



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

(Approved by AICTE | NAAC Accreditation with 'A' Grade | Accredited by NBA | Affiliated to JNTUH)

Dundigal, Hyderabad - 500 043, Telangana

**OUTCOME BASED EDUCATION
WITH
CHOICE BASED CREDIT SYSTEM**

**BACHELOR OF TECHNOLOGY
AERONAUTICAL ENGINEERING**

**ACADEMIC REGULATIONS, COURSE CATALOGUE and SYLLABUS
BT23**

**B.Tech Regular Four Year Degree Program
(for the batches admitted from the academic year 2023 - 2024)**

&

**B.Tech (Lateral Entry Scheme)
(for the batches admitted from the academic year 2024 - 2025)**

**These rules and regulations may be altered / changed from time to time by the academic council
FAILURE TO READ AND UNDERSTAND THE RULES IS NOT AN EXCUSE**

VISION

To bring forth professionally competent and socially sensible engineers, capable of working across cultures meeting the global standards ethically.

MISSION

To provide students with an extensive and exceptional education that prepares them to excel in their profession, guided by dynamic intellectual community and be able to face the technically complex world with creative leadership qualities.

Further, be instrumental in emanating new knowledge through innovative research that emboldens entrepreneurship and economic development for the benefit of wide spread community.

QUALITY POLICY

Our policy is to nurture and build diligent and dedicated community of engineers providing a professional and unprejudiced environment, thus justifying the purpose of teaching and satisfying the stake holders.

A team of well qualified and experienced professionals ensure quality education with its practical application in all areas of the Institute.

PROGRAM OUTCOMES (PO's)

Engineering Graduates will be able to:

- PO1: Engineering Knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- PO2: Problem Analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
- PO3: Design/Development of Solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
- PO4: Conduct Investigations of Complex Problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- PO5: Modern Tool Usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
- PO6: The Engineer and Society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
- PO7: Environment and Sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
- PO8: Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- PO9: Individual and Team Work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- PO10: Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- PO11: Project Management and Finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- PO12: Life-Long Learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

CONTENTS

Section	Particulars	Page
1	Choice Based Credit System	1
2	Medium of Instruction	1
3	Programs Offered	1
4	Semester Structure	2
5	Registration / Dropping / Withdrawal	3
6	Credit System	3
7	Curricular Components	4
8	Evaluation Methodology	5
9	Attendance Requirements and Detention Policy	12
10	Conduct of Semester End Examinations and Evaluation	13
11	Scheme for the Award of Grade	13
12	Letter Grades and Grade Points	13
13	Computation of SGPA and CGPA	14
14	Illustration of Computation of SGPA and CGPA	15
15	Review of SEE Theory Answer Books	16
16	Promotion Policies	16
17	Graduation Requirements	17
18	Award of Degree	17
19	B.Tech with Honours or Minor in Engineering	18
20	Temporary Break of Study from the Program	21
21	Termination from the Program	21
22	With-holding of Results	22
23	Graduation Day	22
24	Discipline	22
25	Grievance Redressal Committee	22
26	Transitory Regulations	22
27	Revision of Regulations and Curriculum	23
28	Frequently asked Questions and Answers about autonomy	24
29	Malpractice Rules	28
30	Course Catalogue	31
31	Undertaking by Student / Parent	453

**“Take up one idea.
Make that one idea your life-think of it, dream of it, live on that idea.
Let the brain muscles, nerves, every part of your body be full of that idea and just leave every other idea
alone. This is the way to success”**

Swami Vivekananda

PRELIMINARY DEFINITIONS AND NOMENCLATURES

AICTE: Means All India Council for Technical Education, New Delhi.

Autonomous Institute: Means an institute designated as Autonomous by University Grants Commission (UGC), New Delhi in concurrence with affiliating University (Jawaharlal Nehru Technological University, Hyderabad) and State Government.

Academic Autonomy: Means freedom to an institute in all aspects of conducting its academic programs, granted by UGC for Promoting Excellence.

Academic Council: The Academic Council is the highest academic body of the institute and is responsible for the maintenance of standards of instruction, education and examination within the institute. Academic Council is an authority as per UGC regulations and it has the right to take decisions on all academic matters including academic research.

Academic Year: It is the period necessary to complete an actual course of study within a year. It comprises two main semesters i.e., (one odd + one even) and one supplementary semester.

Branch: Means specialization in a program like B.Tech degree program in Aeronautical Engineering, B.Tech degree program in Computer Science and Engineering etc.

Board of Studies (BOS): BOS is an authority as defined in UGC regulations, constituted by Head of the Organization for each of the departments separately. They are responsible for curriculum design and updation in respect of all the programs offered by a department.

Backlog Course: A course is considered to be a backlog course, if the student has obtained a failure grade (F) in that course.

Basic Sciences: The courses offered in the areas of Mathematics, Physics, Chemistry etc., are considered to be foundational in nature.

Betterment: Betterment is a way that contributes towards improvement of the students' grade in any course(s). It can be done by either (a) re-appearing or (b) re-registering for the course.

Commission: Means University Grants Commission (UGC), New Delhi.

Choice Based Credit System: The credit based semester system is one which provides flexibility in designing curriculum and assigning credits based on the course content and hours of teaching along with provision of choice for the student in the course selection.

Certificate Course: It is a course that makes a student to have hands-on expertise and skills required for holistic development in a specific area/field.

Compulsory course: Course required to be undertaken for the award of the degree as per the program.

Continuous Internal Examination: It is an examination conducted towards sessional assessment.

Core: The courses that are essential constituents of each engineering discipline are categorized as professional core courses for that discipline.

Course: A course is offered by a department for learning in a particular semester.

Course Outcomes: The essential skills that need to be acquired by every student through a course.

Credit: A credit is a unit that gives weight to the value, level or time requirements of an academic course. The number of 'Contact Hours' in a week of a particular course determines its credit value. One credit is equivalent to one lecture/tutorial hour per week.

Credit point: It is the product of grade point and number of credits for a course.

Cumulative Grade Point Average (CGPA): It is a measure of cumulative performance of a student over all the completed semesters. The CGPA is the ratio of total credit points secured by a student in various courses in all semesters and the sum of the total credits of all courses in all the semesters. It is expressed up to two decimal places.

Curriculum: Curriculum incorporates the planned interaction of students with instructional content, materials, resources, and processes for evaluating the attainment of Program Educational Objectives.

Department: An academic entity that conducts relevant curricular and co-curricular activities, involving both teaching and non-teaching staff, and other resources in the process of study for a degree.

Detention in a Course: Student who does not obtain minimum prescribed attendance in a course shall be detained in that particular course.

Dropping from Semester: Student who doesn't want to register for any semester can apply in writing in prescribed format before the commencement of that semester.

Elective Course: A course that can be chosen from a set of courses. An elective can be Professional Elective and / or Open Elective.

Evaluation: Evaluation is the process of judging the academic performance of the student in her/his courses. It is done through a combination of continuous internal assessment and semester end examinations.

Experiential Engineering Education (ExEEd): Engineering entrepreneurship requires strong technical skills in engineering design and computation with key business skills from marketing to business model generation. Our students require sufficient skills to innovate in existing companies or create their own.

Grade: It is an index of the performance of the students in a said course. Grades are indicated by alphabets.

Grade Point: It is a numerical weight allotted to each letter grade on a 10 - point scale.

Honours: An Honours degree typically refers to a higher level of academic achievement at an undergraduate level.

Institute: Means Institute of Aeronautical Engineering, Hyderabad unless indicated otherwise by the context.

Massive Open Online Courses (MOOC): MOOC courses inculcate the habit of self learning. MOOC courses would be additional choices in all the elective group courses.

Minor: Minor are coherent sequences of courses which may be taken in addition to the courses required for the B.Tech degree.

Pre-requisite: A specific course, the knowledge of which is required to complete before student register another course at the next grade level.

Professional Elective: It indicates a course that is discipline centric. An appropriate choice of minimum number of such electives as specified in the program will lead to a degree with specialization.

Program: Means, UG degree program: Bachelor of Technology (B.Tech); PG degree program: Master of Technology (M.Tech) / Master of Business Administration (MBA).

Program Educational Objectives: The broad career, professional and personal goals that every student will achieve through a strategic and sequential action plan.

Project work: It is a design or research-based work to be taken up by a student during his/her final year to achieve a particular aim. It is a credit based course and is to be planned carefully by the student.

Re-Appearing: A student can reappear only in the semester end examination for theory component of a course to the regulations contained herein.

Registration: Process of enrolling into a set of courses in a semester of a program.

Regulations: The regulations, common to all B.Tech programs offered by Institute, are designated as “BT23” and are binding on all the stakeholders.

Semester: It is a period of study consisting of 16 weeks of academic work equivalent to normally minimum of 90 working days. Odd semester commences usually in July and even semester in December of every year.

Semester End Examinations: It is an examination conducted for all courses offered in a semester at the end of the semester.

S/he: Means “she” and “he” both.

Student Outcomes: The essential skill sets that need to be acquired by every student during her/his program of study. These skill sets are in the areas of employability, entrepreneurial, social and behavioral.

University: Means Jawaharlal Nehru Technological University Hyderabad (JNTUH), Hyderabad, is an affiliating University.

Withdraw from a Course: Withdrawing from a course means that a student can drop from a course within the first two weeks of odd or even semester (deadlines are different for summer sessions). However, s/he can choose a substitute course in place of it, by exercising the option within 5 working days from the date of withdrawal.

PREFACE

Dear Students,

The focus at IARE is to deliver value-based education with academically well qualified faculty and infrastructure. It is a matter of pride that IARE continues to be the preferred destination for students to pursue an engineering degree.

In the year 2015, IARE was granted academic autonomy status by University Grants Commission, New Delhi under Jawaharlal Nehru Technology University Hyderabad. From then onwards, our prime focus is on developing and delivering a curriculum which caters to the needs of various stakeholders. The curriculum has unique features enabling students to develop critical thinking, solve problems, analyze socially relevant issues, etc. The academic cycle designed on the basis of Outcome Based Education (OBE) strongly emphasizes continuous improvement and this has made our curriculum responsive to current requirements.

The curriculum at IARE has been developed by experts from academia and industry and it has unique features to enhance problem solving skills apart from academic enrichment. The curriculum of B.Tech program has been thoroughly revised as per AICTE / UGC / JNTUH guidelines and have incorporated unique features such as competency training / coding, industry driven elective, internship and many more. The curriculum is designed in a way so as to impart engineering education in a holistic approach towards Excellence.

I hope you will have a fruitful stay at IARE.

Dr. L V Narasimha Prasad
Principal



INSTITUTE OF AERONAUTICAL ENGINEERING (Autonomous)

ACADEMIC REGULATIONS – BT23

B.Tech. Regular Four-Year Degree Program
(for the batches admitted from the academic year 2023 - 2024)

&

B.Tech. (Lateral Entry Scheme)
(for the batches admitted from the academic year 2024 - 2025)

For pursuing four year undergraduate Bachelor of Technology (B.Tech) degree program of study in engineering offered by Institute of Aeronautical Engineering under Autonomous status.

A student shall undergo the prescribed courses as given in the program curriculum to obtain his / her degree in major in which he/she is admitted with 160 credits in the entire program of 4 years. Additional 20/18 credits can be acquired for the degree of B.Tech with **Honours or Minor in Engineering**. These additional 20/18 credits will have to be acquired with massive open online courses (MOOCs) / courses offered by the respective department, to tap the zeal and excitement of learning beyond the classrooms. This creates an excellent opportunity for students to acquire the necessary skill set for employability through massive open online courses where the rare expertise of world-famous experts from academics and industry are available.

Separate certificate will be issued in addition to major degree program mentioning that the student has cleared Honours / Minor specialization in respective courses.

1. CHOICE BASED CREDIT SYSTEM

The credit-based semester system provides flexibility in designing program curriculum and assigning credits based on the course content and hours of teaching. The Choice Based Credit System (CBCS) provides a 'cafeteria' type approach in which the students can take courses of their choice, learn at their own pace, undergo additional courses and acquire more than the required credits, and adopt an interdisciplinary approach to learning.

A course defines learning objectives and learning outcomes and comprises lectures / tutorials / laboratory work / field work / project work / seminars / assignments / MOOCs / alternative assessment tools / presentations / self-study etc., or a combination of some of these. Under the CBCS, the requirement for awarding a degree is prescribed in terms of number of credits to be completed by the students.

2. MEDIUM OF INSTRUCTION

The medium of instruction shall be **English** for all courses, examinations, seminar presentations and project work. The program curriculum will comprise courses of study as given in course structure, in accordance with the prescribed syllabi.

3. PROGRAMS OFFERED

Presently, the institute is offering Bachelor of Technology (B.Tech) degree programs in eleven disciplines. The various programs and their two-letter unique codes are given in Table 1.

Table 1: B.Tech Programs offered

S. No	Name of the Program	Title	Code
1	Aeronautical Engineering	AE	07
2	Computer Science and Engineering	CS	05
3	Computer Science and Engineering (AI & ML)	CA	34
4	Computer Science and Engineering (Data Science)	CD	35
5	Computer Science and Engineering (Cyber Security)	CC	36
6	Information Technology	IT	06
7	Electronics and Communication Engineering	EC	04
8	Electrical and Electronics Engineering	EE	02
9	Mechanical Engineering	ME	03
10	Civil Engineering	CE	01

4. SEMESTER STRUCTURE

Each academic year is divided into two semesters, **ODD and EVEN** semester. Both the semesters have regular class work.

- 4.1 Each semester shall be of 21 weeks (Table 2) duration, and this period includes time for course registration, regular class work, examination preparation, and conduction of examinations.
- 4.2 Each semester shall have a minimum of 90 Instructional / working days.
- 4.3 The academic calendar for both Odd and Even semester shown in Table 2 is declared at the beginning of the academic year.

Table 2: Academic Calendar

FIRST SEMESTER (21 weeks)	I Spell Instruction Period	8 weeks	19 weeks
	I Continuous Internal Assessment Examinations (Mid-term)	1 week	
	II Spell Instruction Period	8 weeks	
	II Continuous Internal Assessment Examinations (Mid-term)	1 week	
	Preparation and Practical Examinations	1 week	
	Semester End Examinations		2 weeks
Semester Break and Supplementary Exams			5 weeks
SECOND SEMESTER (21 weeks)	I Spell Instruction Period	8 weeks	19 weeks
	I Continuous Internal Assessment Examinations (Mid-term)	1 week	
	II Spell Instruction Period	8 weeks	
	II Continuous Internal Assessment Examinations (Mid-term)	1 week	
	Preparation and Practical Examinations	1 week	
	Semester End Examinations		2 weeks
Semester Break and Supplementary Exams			5 weeks

- 4.4 Students admitted on transfer from JNTUH affiliated institutes, Universities and other institutes in the courses in which they are required to earn credits so as to be on par with regular students as prescribed by concerned 'Board of Studies'.

5 REGISTRATION / DROPPING / WITHDRAWAL

The academic calendar includes important academic activities to assist the students and the faculty. These include, dates assigned for registration of courses, dropping of courses and withdrawal from courses. This enables the students to be well prepared and take full advantage of the flexibility provided by the credit system.

- 5.1. Each student has to compulsorily register for course work at the beginning of each semester as per the schedule mentioned in the Academic Calendar. It is compulsory for the student to register for courses in time. The registration will be organized departmentally under the supervision of the Head of the department.
- 5.2. In ABSENTIA, registration will not be permitted under any circumstances.
- 5.3. At the time of registration, students should have cleared all the dues of Institute and Hostel for the previous semesters, paid the prescribed fees for the current semester and not been debarred from the institute for a specified period on disciplinary or any other ground.
- 5.4. In the first two semesters, the prescribed course load per semester is fixed and is mandated to register all courses. Withdrawal / dropping of courses in the first and second semester is not allowed.
- 5.5. In all semesters, the average load is 20 credits / semester, with its minimum and maximum limits being set at 16 and 24 credits. This flexibility enables students (**from IV semester onwards**) to cope-up with the course work considering the academic strength and capability of student.

Note: A course may be offered to the students, only if a minimum of 20 students opt for it.

5.6. Dropping of Courses:

Within one week after the last date of first continuous internal examination, the student may in consultation with his / her faculty mentor / adviser, drop one or more courses without prejudice to the minimum number of credits as specified in clause 5.5. The dropped courses are not recorded in the memorandum of grades. Student must complete the dropped course(s) by registering in the forthcoming semester in order to earn the required credits.

5.7. Withdrawal from Courses:

A student is permitted to withdraw from a course before commencement first continuous internal examination. Such withdrawals will be permitted without prejudice to the minimum number of credits as specified in clause 5.5. A student cannot withdraw a course more than once and withdrawal of reregistered courses is not permitted.

6. CREDIT SYSTEM

The B.Tech Program shall consist of a number of courses and each course shall be assigned with credits. The curriculum shall comprise Program Core Courses (PCC), Program Elective Courses (PEC), Open Elective Courses (OEC), Laboratory Courses, Mandatory Courses (MC), Value Added Courses (VAC), Experiential Engineering Education (ExEEd), Internship and Project work.

Depending on the complexity and volume of the course, the number of contact periods per week will be assigned. Each theory and laboratory course carries credits based on the number of hours / week.

- Contact classes (Theory): 1 credit per lecture hour per week, 1 credit per tutorial hour per week.
- Laboratory hours (Practical): 1 credit for 2 practical hours per week.
- Project work: 1 credit for 2 hours of project work per week.

- Experiential Engineering Education (ExEEd): 1 credit for two per hours.
- Mandatory courses / Value added courses : No credit is awarded.

Credit distribution for courses offered is given in Table 3.

Table 3: Credit distribution

S. No	Course	Hours	Credits
1	Theory courses	2 / 3 / 4	2 / 3 / 4
2	Program elective courses / Open elective courses	3 / 2	3 / 2
3	Laboratory courses	2 / 3 / 4	1 / 1.5 / 2
4	Mandatory course / Value added course	-	0
5	Project Work: Phase - I and II	-	14
6	Full Semester Internship (FSI) Project work	-	14

Major benefits of adopting the credit system are listed below:

- Quantification and uniformity in the listing of courses for all programs at institute, like core, electives and project work.
- Ease of allocation of courses under different heads by using their credits to meet national /international practices in technical education.
- Convenience to specify the minimum / maximum limits of course load and its average per semester in the form of credits to be earned by a student.
- Flexibility in program duration for students by enabling them to pace their course load within minimum/maximum limits based on their preparation and capabilities.
- Wider choice of courses available from any department of the same institute or even from other similar institute, either for credit or for audit.
- Improved facility for students to optimize their learning by availing of transfer of credits earned by them from one College to another.

7. CURRICULAR COMPONENTS

Courses in a curriculum may be of three kinds: **Foundation / Skill, Program core courses, Program elective courses and Open elective courses.**

Foundation / Skill Course:

Foundation courses are the courses based upon the content leads to enhancement of skill and knowledge as well as value based and are aimed at man making education. Skill courses are those areas in which one needs to develop a set of skills to learn anything at all. They are fundamental to learning any course.

Program core courses (PCC):

There may be a program core course in every semester. This is the course which is to be compulsorily studied by a student as a core requirement to complete the requirement of a program in the said discipline of study.

Program elective courses (PEC) / Open elective courses (OEC):

Electives provide breadth of experience in respective branch and application areas. The program elective course(s) is a course which can be chosen from a pool of courses. It may be:

- Supportive to the discipline of study
- Providing an expanded scope
- Enabling an exposure to some other discipline / domain
- Nurturing student's proficiency / skill.

An elective may be program elective, is a discipline centric focusing on those courses which add generic proficiency to the students or may be Open elective course, chosen from unrelated disciplines.

There is list of professional elective courses; students can choose not more than two courses from each track. Overall, students can opt for six professional elective courses which suit their project work in consultation with the faculty advisor / mentor. Nevertheless, one course from each of the three open electives has to be selected. A student may also opt for more elective courses in his/her area of interest.

Every course of the B.Tech program will be placed in one of the eight categories with minimum credits as listed in the Table 4.

Table 4: Category Wise Distribution of Credits

S. No	Category	Breakup of Credits
1	Humanities and Social Sciences (HSMC), including Management.	07
2	Basic Science Courses (BSC) including Mathematics, Physics and Chemistry.	28
3	Engineering Science Courses (ESC), including Workshop, Drawing, ExEEd, Basics of Electrical / Electronics / Mechanical / Computer Engineering.	09
4	Program Core Courses (PCC), relevant to the chosen specialization / branch.	75
5	Program Elective Courses (PEC), relevant to the chosen specialization / branch.	18
6	Open Elective Courses (OEC), from other technical and / or emerging course areas.	09
7	Project work (PROJ) / Full Semester Internship (FSI) Project work	14
8	Mandatory Courses (MC) / Value Added Courses (VAC)	Non-Credit
TOTAL		160

Semester wise course break-up

Following are the **TWO** models of course structure out of which any student shall choose or will be allotted with one model based on their academic performance.

- Full Semester Internship (FSI) Model and
- Non Full Semester Internship (NFSI) Model

A student who secures a minimum CGPA of 7.5 upto IV semester with **no current arrears** and maintains the CGPA of 7.5 till VI semester shall be eligible to opt for FSI. Students can opt full semester internship (FSI) either in VII or VIII semesters. In Non-FSI Model, all the selected students shall carry out the course work and project work as specified in the course catalogue.

8. EVALUATION METHODOLOGY

Total marks for each course shall be based on Continuous Internal Assessment (CIA) and Semester End Examinations (SEE). There shall have a uniform pattern of 40:60 for CIA and SEE of both theory and practical courses. The institute shall conduct multiple continuous internal assessments (CIA) for theory courses. All the performances of a student shall be considered for Continuous Internal Assessment (CIA) marks.

Table 5: Outline for Continuous Internal Assessments (CIA-1 and CIA-2) and SEE:

Activities	CIA-1	CIA-2	SEE	Total Marks
Continuous Internal Examination (CIE)	10 marks	10 marks		20 marks
Definitions and Terminology / Quiz	5 marks	5 marks		10 marks
Tech Talk / Assignment	5 marks	5 marks		10 marks
Semester End Examination (SEE)			60 marks	60 marks
Total	--	--	100 marks	

8.1 Continuous Internal Assessments (CIA-1 and CIA-2)

Assessment is an ongoing process that begins with establishing clear and measurable expected outcomes of student learning, provides students with sufficient opportunities to achieve those outcomes, and concludes with gathering and interpreting evidence to determine how well students' learning matches expectations.

The first component (CIA-1) of assessment is for 10 marks, definitions and terminology / quiz carry 05 marks and 05 marks allotted for Tech talk / Assignments. This assessment and score process should be completed after completing of first 50% of syllabus ($2^{1/2}$) of the course/s and within 45 working days of semester program.

The second component (CIA-2) of assessment is for 10 marks, definitions and terminology / quiz carry 05 marks and 05 marks are allotted for Tech talk / Assignments. This assessment and score process should be completed after completing of remaining 50% of syllabus ($2^{1/2}$) of the course/s and within 45 working days of semester program.

In case of a student who has failed to attend the CIA1 or CIA2 on a scheduled date, shall be deemed that the student has dropped the examination. However, in case a student could not take the test on scheduled date due to genuine reasons, may appeal to the HOD / Principal. The HOD / Principal in consultation with the class in-charge shall decide about the genuineness of the case and decide to conduct Make-Up Examination to such candidate on the date fixed by the Examinations Control Office but before commencement of the concerned semester end examinations.

8.2 Definitions and Terminology / Quiz

Definitions and Terminology: The conduction of definitions and terminology is completely online. The faculty should prepare 30 to 35 questions from each and every module by clearly mentioning the answer. The course handling faculty needs to submit detailed document to the respective department.

Mode of conducting examination: Online through e-Examdesk

Quiz: The faculty should prepare 50 multiple choice questions from each module ($50 * 5 = 250$ questions). Each question will have four choices / options, fill in the blanks etc. The course handling faculty needs to submit detailed document to the respective department.

Mode of conducting examination: Online through e-Examdesk

8.3 Tech Talk / Assignment

Tech Talk: Technical talks cover a wide range of technical concepts and ideas. For conduction of Tech Talk faculty has to submit latest topics from IEEE, CSI, magazines etc.

Assignments: The assignments develop different skills and increase their knowledge base significantly. It provides the evidence for the faculty that the students have achieved the goals. It helps the faculty to evaluate the student's understanding of the course. The output can be judged using sensory perception (observing, reading, tasting etc.). Faculty should prepare 30 to 35 assignment questions from each module and submit the same to respective department.

8.4 Semester End Examination (SEE)

The semester end examinations (SEE), for theory courses, will be conducted for 60 marks. The syllabus for the theory courses is divided into FIVE modules and each module carries equal weightage in terms of marks distribution. The question paper has eight questions; module one and two has one question each, and two questions each from modules three, four and five. There will be no choice for first and second modules. From third module onwards there will be an “either” “or” choice, and the student should answer either of the two questions. Each question carries 12 marks and has two parts (A and B). Part A is a descriptive type question for 6 marks and Part B a critical thinking question / problem solving question for 6 marks. **The duration of semester end examination is 3 hours.**

8.5 Passing Criteria:

To maintain high standards in all aspects of examinations at the institute, the institute shall follow the standards of passing at CIA (CIA-1 and CIA-2) and SEE for each course. However, the student's performance in a course shall be judged by taking into account the results of CIE and SEE individually and also together, as shown below:

- a) A minimum of 35% of marks to be secured by cumulating marks in CIA-1 and CIA-2 for appearing for a SEE theory examination.
- b) A minimum of 35% of marks to be scored in SEE for passing a theory course.
- c) A minimum of 40% of marks in CIA+ SEE for passing a theory course.

8.5 Supplementary examinations

Supplementary examinations for the odd semester shall be conducted with the regular examinations of even semester and vice versa. In case of failure in any course, a student may be permitted to register for the same course when offered.

Advanced supplementary examination will be conducted for VIII semester courses at the end of the program after declaration of results.

8.6 Laboratory Course

Evaluation methodology of laboratory course (CIA)

Each laboratory courses there shall be a CIA during the semester for 40 marks and 60 marks for SEE. The 40 marks for internal evaluation marks are awarded as follows:

1. A write-up on day-to-day experiment in the laboratory (in terms of aim, components / procedure, expected outcome) which shall be evaluated for **10 marks**.
2. **10 marks** for viva-voce (or) tutorial (or) case study (or) application (or) poster presentation of the course concerned.
3. Internal practical examination conducted by the laboratory teacher concerned shall be evaluated for **10 marks**.
4. The remaining **10 marks** are for Laboratory Report/Project and Presentation, which consists of the Design (or) Software / Hardware Model Presentation (or) App Development (or) Prototype Presentation which shall be evaluated after completion of laboratory course and before semester end practical examination.

Evaluation methodology of laboratory course (SEE)

The Semester End Examination shall be conducted by an external examiner and the laboratory handling faculty. The external examiner shall be appointed from the other colleges which will be decided by the Principal.

The Semester End Examination held for 3 hours. Total 60 marks are divided and allocated as shown below:

1. 10 marks for write-up
2. 15 for experiment / program

3. 15 for evaluation of results
4. 10 marks for presentation on another experiment/program in the same laboratory course and
5. 10 marks for viva-voce on concerned laboratory course.

8.7 Mandatory Courses (MC)

These courses are among the compulsory courses will not carry any credits. However, a pass in each such course during the program shall be necessary requirement for the student to qualify for the award of degree. No marks or letter grades shall be allotted for mandatory/non-credit courses. Its result shall be declared as **“Satisfactory” or “Not Satisfactory”** performance.

8.8 Additional Mandatory Courses for lateral entry B.Tech students

In addition to the non-credit mandatory courses for regular B.Tech students, the lateral entry students shall take up the following three non-credit mandatory bridge courses (one in III semester, one in IV semester and one in V semester) as listed in Table 6. The student shall pass the following non-credit mandatory courses for the award of the degree and must clear these bridge courses before advancing to the VII semester of the program.

Table-6: Additional Mandatory Courses for lateral entry students

S.No	Additional mandatory courses for lateral entry students
1	Dip-Mathematics
2	Dip-English Communication Skills
3	Dip-Programming for Problem Solving / Essentials of Problem Solving

8.9 Value Added Courses (VAC)

1. Introduction

Value-Added courses are not part of the curriculum and designed to provide necessary skills to increase the employability quotient and equip the students with essential skills to succeed in life.

Institute offers a wide variety of value-added Courses which shall be conducted after class hours. These courses shall be conducted by experts or in-house staff and help students stand apart from the rest in the job market by adding further value to their resume. These value-added courses will be mostly independent to each type of the fields.

2. Objectives

Objectives of the value-added course are:

- Provide students an understanding of the expectations of industry.
- Improve employability skills of students.
- Bridge the skill gaps and make students industry ready.
- Provide an opportunity to students develop their inter-disciplinary skills.
- Mould students as job providers rather than job seekers.

3. Designing the Courses

- Before designing the syllabus, the feedback from the employers, alumni and industry people will be analyzed and considered to select and design an appropriate course by identifying the gaps and also understand the expectations for current and emerging trends.
- Any new value-added course developed by a department should be placed before the Board of Studies for approval.
- The course offered should not be the same as any course listed in the curriculum of the respective Program / or any other program offered in the institute.
- A unique course code is to be given for each course.

4. Guidelines for conducting value added courses

- Value Added Course is not mandatory to qualify for any program.
- It is a teacher assisted learning course open to all students without any additional fee.
- Classes for VAC will be conducted beyond the regular class work only.
- A student will be permitted to register only one value added course in a semester.

5. Duration and Venue

- The duration of value-added course should not be less than 30 hours.
- The respective Head of the department shall provide class room/s based on the number of students/batches.

6. Procedure for Registration:

The list of value-added courses shall be displayed in the institute website along with the syllabus, objectives and outcomes. A student shall register for a value-added course offered during the semester by submitting the duly filled in registration form. The Head of the department shall segregate the list of students enrolled for the value-added course and submit the details to Dean of academics before the start of course.

7. Attendance

Value added course handling faculty shall be responsible for the maintenance of attendance and assessment who have registered for the course.

- The record shall contain details of the students' attendance, number of classes attended and also Record shall also contain the organisation of lesson plan of the Course Instructor.
- The record shall be submitted to the Head of the department for monitoring the attendance.
- At the end of the semester, the record shall be duly signed by the course coordinator and the Head of the Department and placed in safe custody for any future verification.
- Each student shall have a minimum of 75% attendance in all the courses of the particular semester failing which he or she will not be permitted to write the semester end examination.

8. Passing Requirement and Grading

- The passing requirement for value added courses shall be 40% of the marks prescribed for the course.
- A candidate who has not secured a minimum of 40% of marks in a course (internal and end-term) shall reappear for the course in the next semester/year.
- The grades obtained in value-added courses will not be included for calculating the CGPA.

9. Course Completion

- Students will get a certificate after they have registered for, written the exam and successfully passed.
- The students who have successfully completed the value-added Course shall be issued with a Certificate duly signed by the Authorized signatories.

Note: Apart from the above, students can also register and get the value-added course completion certificate by registering the courses from SWAYAM, e-PG patashala (NPTEL).

8.10 Experiential Engineering Education (ExEED)

Engineering entrepreneurship requires strong technical skills in engineering design and computation with key business skills from marketing to business model generation. Students require sufficient skills to innovate in existing companies or create their own.

This course will be evaluated for a total of 100 marks consisting of 40 marks for internal assessment and 60 marks for semester end Examination. Out of 40 marks of internal assessment, students has to submit Innovative Idea in a team of four members in the given format. The semester end examination for 60 marks shall be conducted internally, students has to present the Innovative Idea and it will be evaluated by

internal ExEEd faculty with at least one faculty member as examiner from the industry, both nominated by the Principal from the panel of experts recommended by the Dean-CLET.

Center for Outcome Based Teaching and Learning (OBTL) of the institute design and teach ExL power skills courses, to shape the student's future. All the below mentioned Experiential Engineering Education (ExEED) courses are evaluated for one credit each.

- **ExL – Essential of Innovation:** This course creates platform where students experience a hands-on approach to learning about engineering. It focuses on educating the students about diversified platforms for learning the skills, career development, innovations, entrepreneurship etc. Based on the requirements this course is offered in first or second semester.
- **ExL – Prototype / Model Development:** It covers the application of relevant technologies to create interaction prototypes. Students learn about different kinds of prototyping activities involved in designing low-fidelity and high-fidelity prototypes such as POC models, web pages and mobile interfaces etc. This course introduces key concepts, processes and principles of industry driven digital fabrication in a manufacturing environment. Students will undertake small-scale, team-based project work to create fabricated objects that relate to a local industry, organisation or community need or opportunity.

8.11 Project Work / FSI Project Work

This gives students a platform to experience a research driven career in engineering, while developing a device / systems and publishing in reputed SCI / SCOPUS indexed journals and/or filing an **Intellectual Property** (IPR-Patent/Copyright) to aid communities around the world. Students should work individually as per the guidelines issued by head of the department concerned. The benefits to students of this mode of learning include increased engagement, fostering of critical thinking and greater independence.

The topic should be so selected that the students are enabled to complete the work in the stipulated time with the available resources in the respective laboratories. The scope of the work be handling part of the consultancy work, maintenance of the existing equipment, development of new experiment setup or can be a prelude to the main project with a specific outcome.

Project report will be evaluated for 100 marks in total. Assessment will be done for 100 marks out of which, the supervisor / guide will evaluate for 40 marks based on the work and presentation / execution of the work. Subdivision for the remaining 60 marks is based on publication, report, presentation, execution and viva-voce. Evaluation shall be done by a committee comprising the supervisor, Head of the department, and an examiner nominated by the Principal.

8.12 Skill enhancement project

Students must submit the skill enhancement project report of the specified course which are included in the course catalogue. If the student has failed to submit the report or not reached upto the mark, needs to re-register the course in next semesters till completion.

8.13 Project work

The student's project activity is spread over in VII semester and in VIII semesters. A student shall carry out the project work under the supervision of a faculty member or in collaboration with an Industry, R&D organization or another academic institution / University where sufficient facilities exist to carry out the project work.

Project work (Phase - I) starts in VII semester as it takes a vital role in campus hiring process. Students shall select project titles from their respective logins uploaded by the supervisors at the middle of the VI semester. Two reviews are conducted by department review committee (DRC). Student must submit a project report summarizing the work done up to design phase/prototype by the end of VII semester. The semester end examination for project work (Phase-I) is evaluated based on the project report submitted and a viva-voce exam for 60 marks by a committee comprising the head of the department, the project supervisor and an external examiner nominated by the Principal.

Project Work (Phase - II) starts in VIII semester, and it shall be evaluated for 100 marks out of which 40 marks towards continuous internal assessment and 60 marks for semester end examination. Two reviews are to be conducted by DRC on the progress of the project for 40 marks. The semester end examination shall be based on the final report submitted and a viva-voce exam for 60 marks by a committee comprising the head of the department, the project supervisor and an external examiner nominated by the Principal.

A minimum of 50% of maximum marks shall be obtained to earn the corresponding credits.

8.14 Full Semester Internship (FSI)

FSI is a full semester internship program carry 14 credits. The FSI shall be opted in VII semester or in VIII semester. During the FSI, student has to spend one full semester in an identified industry / firm / R&D organization or another academic institution/University where sufficient facilities exist to carry out the project work.

Following are the evaluation guidelines:

- Quizzes: 2 times
- Quiz #1 - About the industry profile, weightage: 5%
- Quiz #2 - Technical-project related, weightage: 5%
- Seminars - 2 times (once in six weeks), weightage: 7.5% + 7.5%
- Viva-voce: 2 times (once in six weeks), weightage: 7.5% + 7.5%
- Project Report, weightage: 15%
- Internship Diary, weightage: 5 %
- Final Presentation, weightage: 40%

FSI shall be open to all the branches with a ceiling of maximum 10% distributed in both semesters. The selection procedure is:

- Choice of the students.
- CGPA (> 7.5) upto IV semester having no credit arrears.
- Competency Mapping / Allotment.

It is recommended that the FSI Project work leads to a research publication in a reputed Journal / Conference or the filing of patent / design with the patent office, or, the start-up initiative with a sustainable and viable business model accepted by the incubation center of the institute together with the formal registration of the startup.

8.15 Field projects / Internship Academic attachment:

The Field Projects (FP) / Internships are mandatory for the students admitted from the academic year 2023 -24 onwards. It is spreaded over from II semester to VI semester.

Field Project: Field project (FP) integrates theory and practice by providing students with an opportunity to work on real-world challenges. It can be used to learn about the functioning and manufacturing procedures of a factory. Besides this, student can also learn about the geographical factors of the region for the specific products /equipment.

Internship is an integral part of the academic curriculum, it is a learning activity in which a student fortifies and deepens his/her theoretical knowledge and skills attained in the classrooms by integrating with practical activities. It offers the students an opportunity to gain hands-on industrial or organizational exposure; to integrate the knowledge and skills acquired through the coursework; interact with professionals and other interns; and to improve their presentation, writing, and communication skills. Internship often acts as a gateway for final placement for many students.

Table 7: Possibility of availing opportunities during semester breaks.

S.No	Schedule	Duration	Type
1	At the end of II semester / Before commencement of III semester	2 Weeks	Field Project
2	At the end of IV semester / Before commencement of V semester	2 Weeks	Internship
3	At the end of VI semester / Before commencement of VII semester	2 Weeks	Internship

Evaluation Methodology of Field Project / Internships:

The evaluation of the field project / field practicum / Internships will be done before commencement of subsequent semester specified in Table: 7. The students have to submit a detailed report of field project / field practicum / Internships through online portal and also carry hard copy of report with geo-tagged photographs. The committee will evaluate by enclosing their comments like **satisfactory or not satisfactory**. If students get not satisfactory results, reports need to be re-submit in the respective department once again for evaluation.

8.16 Plagiarism index for Project report:

All project reports shall go through the plagiarism check and the plagiarism index has to be less than 20%. Project reports with plagiarism more than 20% and less than 60% shall be asked for resubmission within a stipulated period of six months. Project reports with plagiarism more than 60% shall be rejected.

9. ATTENDANCE REQUIREMENTS AND DETENTION POLICY

- 9.1 A student shall be eligible to appear for the semester end examinations, if the student acquires a minimum of 75% of attendance in aggregate of all the courses (including attendance in mandatory courses like Environmental Science, Constitution of India, and Gender Sensitization etc..) for that semester. **Two periods** of attendance for each theory course shall be considered, if the student appears for the mid-term examination of that course.
- 9.2 Shortage of attendance in aggregate upto 10% (65% and above, and below 75%) in each semester may be condoned by the college academic committee on genuine and valid grounds, based on the student's representation with supporting evidence.
- 9.3 A stipulated fee shall be payable for condoning of shortage of attendance.
- 9.4 Shortage of attendance below 65% in aggregate shall in NO case be condoned.
- 9.5 Students whose shortage of attendance is not condoned in any semester are not eligible to take their semester end examinations of that semester. They get detained and their registration for that semester shall stand cancelled, including all academic credentials (internal marks etc.) of that semester. They will not be promoted to the next semester. They may seek re-registration for all those courses registered in that semester in which the student is detained, by seeking re-admission into that semester as and when offered; if there are any program electives and / or open electives, the same may also be re- registered if offered. However, if those electives are not offered in later semesters, then alternate electives may be chosen from the **same** set of elective courses offered under that category.
- 9.6 A student fulfilling the attendance requirement in the present semester shall not be eligible for readmission into the same class.
- 9.7 A student detained in a semester due to shortage of attendance may be re-admitted in the same semester in the next academic year for fulfillment of academic requirements. The academic regulations under which a student has been re-admitted shall be applicable. Further, no grade allotments or SGPA/ CGPA calculations will be done for the entire semester in which the student has been detained.
- 9.8 A student detained due to lack of credits, shall be promoted to the next academic year only after acquiring the required number of academic credits. The academic regulations under which the student has been readmitted shall be applicable to him.

10. CONDUCT OF SEMESTER END EXAMINATIONS AND EVALUATION

- 10.1 Semester end examination shall be conducted by the Controller of Examinations (COE) by inviting question papers from the external examiners.

- 10.2 COE shall invite 3 - 9 internal / external examiners to evaluate all the semester end examination answer books on a prescribed date(s). Practical laboratory examinations are conducted involving external examiners.
- 10.3 Examinations control office shall consolidate the marks awarded by examiner/s and award the grades.

11. SCHEME FOR THE AWARD OF GRADE

- 11.1 A student shall be deemed to have satisfied the minimum academic requirements and earn the credits for each theory course, if s/he secures,
- Not less than 35% marks for each theory course in the semester end examination, and
 - A minimum of 40% marks for each theory course considering Continuous Internal Assessment (CIA) and Semester End Examination (SEE).
- 11.2 A student shall be deemed to have satisfied the minimum academic requirements and earn the credits for each Laboratory / Project work / FSI Project work, if s/he secures,
- Not less than 40% marks for each Laboratory / Project work / FSI project work course in the semester end examination,
 - A minimum of 40% marks for each Laboratory / Project work / FSI project work course considering both internal and semester end examination.
- 11.3 If a candidate fails to secure a pass in a particular course, it is mandatory that s/he shall register and reappear for the examination in that course during the next semester when examination is conducted in that course. It is mandatory that s/he should continue to register and reappear for the examination till s/he secures a pass.
- 11.4 A student shall be declared successful or 'passed' in a semester, if he secures a Grade Point ≥ 5 ('C' grade or above) in every course in that semester (i.e. when the student gets an SGPA ≥ 5.0 at the end of that particular semester); and he shall be declared successful or 'passed' in the entire under graduate programme, only when gets a CGPA ≥ 5.0 for the award of the degree as required.

12. LETTER GRADES AND GRADE POINTS

- 12.1 Performances of students in each course are expressed in terms of marks as well as in Letter Grades based on absolute grading system. The UGC recommends a 10-point grading system with the following letter grades as given in the Table 8.

Table-8: Grade Points Scale (Absolute Grading)

% of Marks Secured in a Course (Class Intervals)	Letter Grade	Grade Point
Greater than or equal to 90%	O (Outstanding)	10
80 and less than 90%	A+ (Excellent)	9
70 and less than 80%	A (Very Good)	8
60 and less than 70%	B+ (Good)	7
50 and less than 60%	B (Average)	6
40 and less than 50%	C (Pass)	5
Below 40%	F (Fail)	0
Absent	AB (Absent)	0

- 12.2 A student is deemed to have passed and acquired to correspondent credits in particular course if s/he obtains any one of the following grades: "O", "A+", "A", "B+", "B", "C".

- 12.3 A student obtaining Grade F shall be considered Failed and will be required to reappear in the examination.
- 12.4 For non credit courses, 'PP' or "NP" is indicated instead of the letter grade and this will not be counted for the computation of SGPA / CGPA.
- 12.5 At the end of each semester, the institute issues grade sheet indicating the SGPA and CGPA of the student. However, grade sheet will not be issued to the student if s/he has any outstanding dues.

Table 09: Percentage Equivalence of Grade Points (for a 10 – Point Scale)

Grade Point	Percentage of Marks / Class
5.5	50
6.0	55
6.5	60
7.0	65
7.5	70
8.0	75

Note:

- (1) The following Formula for Conversion of CGPA to percentage of marks to be used only after a student has successfully completed the program:

$$\text{Percentage of Marks} = (\text{CGPA} - 0.5) \times 10$$

- (2) Class designation:

≥75% (First Class with Distinction),

≥ 60% and <75 % (First Class),

<60 % and ≥50% (Second Class),

≥40% and <50% (Pass Class).

- (3) The SGPA will be computed and printed on the Memorandum of Grades only if the candidate passes in all the courses offered and gets minimum C grade in all the courses.
- (4) CGPA is calculated only when the candidate passes in all the courses offered in all the semesters.

13. COMPUTATION OF SGPA AND CGPA

The UGC recommends to compute the Semester Grade Point Average (SGPA) and Cumulative Grade Point Average (CGPA). The credit points earned by a student are used for calculating the Semester Grade Point Average (SGPA) and the Cumulative Grade Point Average (CGPA), both of which are important performance indices of the student. SGPA is equal to the sum of all the total points earned by the student in a given semester divided by the number of credits registered by the student in that semester. CGPA gives the sum of all the total points earned in all the previous semesters and the current semester divided by the number of credits registered in all these semesters. Thus,

$$SGPA = \sum_{i=1}^n (C_i G_i) / \sum_{i=1}^n C_i$$

Where, C_i is the number of credits of the i^{th} course and G_i is the grade point scored by the student in the i^{th} course and n represent the number of courses in which a student is registered in the concerned semester.

$$CGPA = \sum_{j=1}^m (C_j S_j) / \sum_{j=1}^m C_j$$

Where, S_j is the SGPA of the j^{th} semester and C_j is the total number of credits upto the semester and m represent the number of semesters completed in which a student registered upto the semester.

The SGPA and CGPA shall be rounded off to 2 decimal points and reported in the transcripts.

14.0 ILLUSTRATION OF COMPUTATION OF SGPA AND CGPA

14.1 Illustration for SGPA

Course Name	Course Credits	Grade letter	Grade point	Credit Point (Credit x Grade)
Course 1	4	A	8	$4 \times 8 = 32$
Course 2	4	O	10	$4 \times 10 = 40$
Course 3	4	C	5	$4 \times 5 = 20$
Course 4	3	B	6	$3 \times 6 = 18$
Course 5	3	A+	9	$3 \times 9 = 27$
Course 6	3	C	5	$3 \times 5 = 15$
	21			152

Thus, $SGPA = 152 / 21 = 7.24$

14.2 Illustration for calculation of CGPA upto 3rd semester

Semester	Course Title	Credits Allotted	Letter Grade Secured	Corresponding Grade Point (GP)	Credit Points (CP)
I	Course 1	3	A	8	24
I	Course 2	3	O	10	30
I	Course 3	3	B	6	18
I	Course 4	4	A	8	32
I	Course 5	3	A+	9	27
I	Course 6	4	C	5	20
II	Course 7	4	B	6	24
II	Course 8	4	A	8	32
II	Course 9	3	C	5	15
II	Course 10	3	O	10	30
II	Course 11	3	B+	7	21
II	Course 12	4	B	6	24
II	Course 13	4	A	8	32
II	Course 14	3	O	10	30
III	Course 15	2	A	8	16
III	Course 16	1	C	5	5
III	Course 17	4	O	10	40
III	Course 18	3	B+	7	21
III	Course 19	4	B	6	24
III	Course 20	4	A	8	32
III	Course 21	3	B+	7	21
	Total Credits	69		Total Credits	518

Thus, $CGPA = 518 / 69 = 7.51$

- The calculation process of CGPA illustrated above will be followed for each subsequent semester until 8th semester. The CGPA obtained at the end of 8th semester will become the final CGPA secured for entire B.Tech. program.
- For merit ranking or comparison purposes or any other listing, only the rounded off^o values of the CGPAs will be used.
- SGPA and CGPA of a semester will be mentioned in the semester Memorandum of Grades if all

courses of that semester are passed in first attempt, otherwise the SGPA and CGPA shall be mentioned only on the Memorandum of Grades in which sitting s/he passed his last exam in that semester. However, mandatory courses will not be taken into consideration.

15. REVALUATION

If the examinee is not satisfied with the marks awarded, s/he may apply for revaluation of answer book in prescribed format online within three (3) working days from the date of declaration of result of the examination or issue of the statement of marks, whichever is earlier. The revaluation facility shall be for theory papers only. The revaluation of answer book shall not be permitted in respect of the marks awarded to the scripts of practical examination / project work (including theory part) and in viva voce / oral / comprehensive examinations.

The re-evaluation will be done by a second independent examiner. The result after re-evaluation shall be as follows:

1. The revaluation marks are considered only if the difference between the original award and award on reevaluation is more than equal to 15% of 60 marks (09 marks).
2. If the difference between the original award and the award on reevaluation is more than 20% (12 marks), a third evaluator is to be appointed and the average of two nearest awards (in the range of 15%) shall be considered.

16. PROMOTION POLICIES

The following academic requirements have to be satisfied in addition to the attendance requirements mentioned in item no. 9.

16.1 For students admitted into B.Tech (Regular) program

- 16.1.1 A student will not be promoted from II semester to III semester unless s/he fulfills the academic requirement of securing 50% of the total credits (rounded to the next lowest integer) from I and II semester examinations, whether the candidate takes the examination(s) or not.
- 16.1.2 A student will not be promoted from IV semester to V semester unless s/he fulfills the academic requirement of securing 60% of the total credits (rounded to the next lowest integer) upto III semester **or** 60% of the total credits (rounded to the next lowest integer) up to IV semester, from all the examinations, whether the candidate takes the examination(s) or not.
- 16.1.3 A student shall be promoted from VI semester to VII semester only if s/he fulfills the academic requirements of securing 60% of the total credits (rounded to the next lowest integer) up to V semester **or** 60% of the total credits (rounded to the next lowest integer) up to VI semester from all the examinations, whether the candidate takes the examination(s) or not.
- 16.1.4 A student shall register for all the 160 credits and earn all the 160 credits. Marks obtained in all the 160 credits shall be considered for the award of the Grade.

16.2 For students admitted into B.Tech (Lateral entry students)

- 16.2.1 A student will not be promoted from IV semester to V semester unless s/he fulfills the academic requirement of securing 60% of the total credits (rounded to the next lowest integer) up to IV semester, from all the examinations, whether the candidate takes the examination(s) or not.
- 16.2.2 A student shall be promoted from VI semester to VII semester only if s/he fulfills the academic requirements of securing 60% of the total credits (rounded to the next lowest integer) up to V semester **or** 60% of the total credits (rounded to the next lowest integer) up to VI semester from all the examinations, whether the candidate takes the examination(s) or not.
- 16.2.3 A student shall register for all the 120 credits and earn all the 120 credits. Marks obtained in all the 120 credits shall be considered for the award of the Grade.

17. GRADUATION REQUIREMENTS

The following academic requirements shall be met for the award of the B.Tech degree.

- 17.1 Student shall register and acquire minimum attendance in all courses and secure 160 credits (with minimum CGPA of 5.0), for regular program and 120 credits (with minimum CGPA of 5.0), for lateral entry program. **There is NO exemption of credits in any case.**
- 17.2 A student of a regular program, who fails to earn 160 credits within **eight consecutive academic years** from the year of his/her admission with a minimum CGPA of 5.0, shall forfeit his/her degree and his/her admission stands cancelled.
- 17.3 A student of a lateral entry program who fails to earn 120 credits within **six consecutive academic years** from the year of his/her admission with a minimum CGPA of 5.0, shall forfeit his/her degree and his/her admission stands cancelled.

18. AWARD OF DEGREE

18.1 Classification of degree will be as follows:

CGPA > 8.0	CGPA \geq 7.0 and < 8.0	CGPA \geq 6.0 and < 7.0	CGPA \geq 5.0 and < 6.0	CGPA < 5.0
First Class with Distinction	First Class	Second Class	Pass Class	Fail

- 18.2 A student with final CGPA (at the end of the under graduate programme) >8.0 , and fulfilling the following conditions - shall be placed in '**first class with distinction**'. However,
- Should have passed all the courses in '**first appearance**' within the first 4 academic years (or 8 sequential semesters) from the date of commencement of first semester.
 - Should have secured a CGPA >8.0 , at the end of each of the 8 sequential semesters, starting from first semester onwards.
 - Should not have been detained or prevented from writing the semester end examinations in any semester due to shortage of attendance or any other reason.
 - A student not fulfilling any of the above conditions with final CGPA >8.0 shall be placed in '**first class**'.
- 18.3 Students with final CGPA (at the end of the B.Tech program) ≥ 7.0 but <8.0 shall be placed in '**first class**'.
- 18.4 Students with final CGPA (at the end of the B.Tech program) ≥ 6.0 but <7.0 , shall be placed in '**second class**'.
- 18.5 All other students who qualify for the award of the degree (as per item 18), with final CGPA (at the end of the B.Tech program) ≥ 5.0 but <6.0 , shall be placed in '**pass class**'.
- 18.6 A student with final CGPA (at the end of the B.Tech program) <5.0 will not be eligible for the award of the degree.
- 18.7 Students fulfilling the conditions listed under item 18.2 alone will be eligible for award of '**Gold Medal**'.

All the candidates who register for the semester end examination will be issued a memorandum of grades sheet by the institute. Apart from the semester wise memorandum of grades sheet, the institute will issue the provisional certificate and consolidated grades memorandum course to the fulfillment of all the academic requirements.

19. B.TECH WITH HONOURS OR MINOR IN ENGINEERING

Students acquiring 160 credits are eligible to get B.Tech degree in Engineering. A student will be eligible to get B.Tech degree with Honours or Minor in Engineering, if s/he completes an additional 20/18 credits (3/4 credits per course). These could be acquired through MOOCs from SWAYAM / NPTEL only. The list for MOOCs will be a dynamic one, as new courses are added from time to time. Few essential skill sets required for employability are also identified year wise. Students interested in doing MOOC courses shall register the course title at their department office at the start of the semester against the courses that are announced by the department. Any expense incurred for the MOOC course / summer program should be met by the students.

Students having no credit arrears and a CGPA of 7.5 or above at the end of the fourth semester are eligible to register for B.Tech (Honours / Minor). After registering for the B.Tech (Honours / Minor) program, if a student fails in any course, s/he will not be eligible for B.Tech (Honours / Minor).

Every department should develop and submit a Honours / Minor – courses list of 5 - 6 theory courses, laboratory and project work.

Honours Certificate for Vertical in his/her OWN Branch for Research orientation; Minor in any other branch for Improving Employability.

Honours will be reflected in the degree certificate as “B.Tech (Honours) in XYZ Engineering”. Similarly, Minor as “B.Tech in XYZ Engineering with Minor in ABC”.

19.1. B.Tech with Honours

The key objectives of offering B. Tech. with Honors program are:

- To expand the domain knowledge of the students laterally and vertically.
- To increase the employability of undergraduate students with expanded knowledge in one of the core engineering disciplines.
- To provide an opportunity to students to pursue their higher studies in wider range of specializations.

Academic Regulations for B. Tech. Honours degree

1. The weekly instruction hours, internal and external evaluation and award of grades are on par with regular 4-Years B. Tech. program.
2. For B. Tech with Honors program, a student needs to earn additional 20 credits (over and above the required 160 credits for B. Tech degree). All these 20 credits required to be attained for B.Tech Honors degree credits are distributed from V semester to VII semester.
3. After registering for the Honors program, if a student is unable to pass all courses in first attempt and earn the required 20 credits, he/she shall not be awarded Honours degree. However, if the student earns all the required 160 credits of B. Tech., he/she will be awarded only B. Tech degree in the concerned branch.
4. There is no transfer of credits from courses of Honors program to regular B. Tech. degree course & vice versa.
5. These **20 credits** are to be earned from the additional courses offered by the host department in the institute or from closely related departments in the institute as well as from the MOOCS platform (NPTEL only).
6. The choice to opt/take the Honors program is purely on the choice of the students.
7. The student shall be given a choice of withdrawing all the courses registered and/or the credits earned for Honours program at any time; and in that case the student will be awarded only B.Tech. degree in the concerned branch on earning the required credits of 160.
8. The students of every branch can choose Honours program in their respective branches if they are eligible for the Honors program. A student who chooses an **Honors program is not eligible to choose a Minor program and vice-versa.**
9. A student can graduate with Honors if he/she fulfils the requirements for his/her regular B.Tech. program as well as fulfills the requirements for Honours program.
10. The institute shall maintain a record of students registered and pursuing their Honors programs branch-wise.
11. The department shall prepare the time-tables for each Honors program offered at their respective departments without any overlap/clash with other courses of study in the respective semesters.

Eligibility conditions of the students for the B.Tech Honors degree

- a) A student can opt for B.Tech. degree with Honors, if she/he passed all courses in first attempt in all the semesters till the results announced and maintaining **7.5 or more CGPA.**
- b) If a student fails in any registered course of either B.Tech or Honours in any semester of four years

program, he/she will not be eligible for obtaining Honors degree. He will be eligible for only

- c) Prior approval of mentor and Head of the Department for the enrolment into Honors program, before commencement of V Semester, is mandatory.
- d) If more than **30% of the students** in a branch fulfill the eligibility criteria (as stated above), the number of students given eligibility should be limited to 30%. The criteria to be followed for choosing 30% candidates in a branch may be the CGPA secured by the students till **III** semester.
- e) Successful completion of **20 credits** earmarked for Honours program with at least 7.5 CGPA along with successful completion of 160 credits earmarked for regular B. Tech. Program with at least 7.5 CGPA and passing all courses in first attempt gives the eligibility for the award of B.Tech (Honors) degree.
- f) For CGPA calculation of B.Tech. course, the **20 credits** of Honours program will not be considered.

Following are the details of such Honors which include some of the most interesting areas in the profession today:

S. No	Department	Honors scheme
1	Aeronautical Engineering	Aerospace Engineering / Space Science etc.
2	Computer Science and Engineering / Information Technology	Big data and Analytics / Cyber Physical Systems, Information Security / Cognitive Science / Artificial Intelligence/ Machine Learning / Data Science / Internet of Things (IoT) / Cyber Security etc.
3	Electronics and Communication Engineering	Digital Communication / Signal Processing / Communication Networks / VLSI Design / Embedded Systems etc.
4	Electrical and Electronics Engineering	Renewable Energy systems / Energy and Sustainability / IoT Applications in Green Energy Systems etc.
5	Mechanical Engineering	Industrial Automation and Robotics / Manufacturing Sciences and Computation Techniques etc.
6	Civil Engineering	Structural Engineering / Environmental Engineering etc.

19.2 B.Tech with Minor in Engineering

The key objectives of offering B.Tech with Minor program are:

- To expand the domain knowledge of the students in one of the other branches of engineering.
- To increase the employability of undergraduate students keeping in view of better opportunity in interdisciplinary areas of engineering & technology.
- To provide an opportunity to students to pursue their higher studies in the inter-disciplinary areas in addition to their own branch of study.
- To offer the knowledge in the areas which are identified as emerging technologies/thrust areas of Engineering.

Advantages of Minor in Engineering

The minors mentioned above are having lots of advantages and a few are listed below:

- To enable students to pursue allied academic interest in contemporary areas.
- To provide an academic mechanism for fulfilling multidisciplinary demands of industries.
- To provide effective yet flexible options for students to achieve basic to intermediate level competence in the Minor area.
- Provides an opportunity to students to become entrepreneurs and leaders by taking business/management minor.
- Combination in the diverse fields of engineering e.g., CSE (Major) + Electronics (Minor) combination increases placement prospects in chip designing companies.
- Provides an opportunity for applicants to pursue higher studies in an inter-disciplinary field of study.
- To increase the overall scope of the undergraduate degrees.

Academic Regulations for B.Tech Degree with Minor programs

1. The weekly instruction hours, internal & external evaluation and award of grades are on par with regular 4-Years B. Tech. program.
2. For B. Tech. with Minor, a student needs to earn additional 18 credits (over and above the required 160 credits for B. Tech degree). The courses are offered from V semester to VII semester only, to obtain minor degree students required to obtain 18 credits.
3. After registering for the Minor program, if a student is unable to earn all the required 18 credits in a specified duration (twice the duration of the course), he/she shall not be awarded Minor degree. However, if the student earns all the required 160 credits of B.Tech, he/she will be awarded only B. Tech degree in the concerned branch.
4. There is no transfer of credits from Minor program courses to regular B. Tech. degree course & vice versa.
5. These 18 credits are to be earned from the additional courses offered by the host department in the institute as well as from the MOOCs platform.
6. For the course selected under MOOCs platform (NPTEL) following guidelines may be followed:
 - a) Prior to registration of MOOCs courses, formal approval of the courses, by the institute is essential, before the issue of approval considers the parameters like the institute / agency which is offering the course, syllabus, credits, duration of the program and mode of evaluation etc.
 - b) Minimum credits for MOOCs course must be equal to or more than the credits specified in the Minor course structure provided by the institute.
 - c) Only Pass-grade / marks or above shall be considered for inclusion of grades in minor grade memo.
 - d) Any expenses incurred for the MOOCs courses are to be met by the students only.
7. The choice to opt / take a Minor program is purely on the choice of the students.
8. The student shall be given a choice of withdrawing all the courses registered and / or the credits earned for Minor program at any time; and in that case the student will be awarded only B. Tech. degree in the concerned branch on earning the required credits of 160.
9. The student can choose only one Minor program along with his / her basic engineering degree. A student who chooses an **Honors program is not eligible to choose a Minor program and vice-versa.**
10. The institute shall maintain a record of students registered and pursuing their Minor programs, minor program-wise and parent branch-wise.
11. The institute / department shall prepare the time-tables for each Minor course offered at their respective institutes without any overlap/clash with other courses of study in the respective semesters.

Eligibility conditions for the student to register for Minor course

- a) A student can opt for B.Tech. degree with Minor program if she/he has no active backlogs till III semester at the time of entering into III year I semester.
- b) Prior approval of mentor and Head of the Department for the enrolment into Minor program, before commencement of III year I Semester (V Semester), is mandatory.
- c) If more than 50% of the students in a branch fulfill the eligibility criteria (as stated above), the number of students given eligibility should be limited to 50%.

20.0 TEMPORARY BREAK OF STUDY FROM THE PROGRAM

- 20.1 A candidate is normally not permitted to take a break from the study. However, if a candidate intends to temporarily discontinue the program in the middle for valid reasons (such as accident or hospitalization due to prolonged ill health) and to rejoin the program in a later respective semester, s/he shall seek the approval from the Principal in advance. Such application shall be submitted

before the last date for payment of examination fee of the semester and forwarded through the Head of the Department stating the reasons for such withdrawal together with supporting documents and endorsement of his / her parent / guardian.

- 20.2 The institute shall examine such an application and if it finds the case to be genuine, it may permit the student to temporarily withdraw from the program. Such permission is accorded only to those who do not have any outstanding dues / demand at the College / University level including tuition fees, any other fees, library materials etc.
- 20.3 The candidate has to rejoin the program after the break from the commencement of the respective semester as and when it is offered.
- 20.4 The total period for completion of the program reckoned from the commencement of the semester to which the candidate was first admitted shall not exceed the maximum period specified in clause 17. The maximum period includes the break period.
- 20.5 If any candidate is detained for any reason, the period of detention shall not be considered as 'Break of Study'.

21. TERMINATION FROM THE PROGRAM

The admission of a student to the program may be terminated and the student is asked to leave the institute in the following circumstances:

- a. The student fails to satisfy the requirements of the program within the maximum period stipulated for that program.
- b. A student shall not be permitted to study any semester more than three times during the entire program of study.
- c. The student fails to satisfy the norms of discipline specified by the institute from time to time.

22. WITH-HOLDING OF RESULTS

If the candidate has not paid any dues to the institute / if any case of indiscipline / malpractice is pending against him, the results and the degree of the candidate will be withheld.

23. GRADUATION DAY

The institute shall have its own annual Graduation Day for the award of degrees to the students completing the prescribed academic requirements in each case, in consultation with the University and by following the provisions in the Statute. The college shall institute prizes and medals to meritorious students and award them annually at the Graduation Day. This will greatly encourage the students to strive for excellence in their academic work.

24. DISCIPLINE

Every student is required to observe discipline and decorum both inside and outside the institute and are expected not to indulge in any activity which will tend to bring down the honour of the institute. If a student indulges in malpractice in any of the theory / practical examination, continuous assessment examinations, he/she shall be liable for punitive action as prescribed by the institute from time to time.

25. GRIEVANCE REDRESSAL COMMITTEE

The institute shall form a Grievance Redressal Committee for each course in each department with the Course Teacher and the HOD as the members. The committee shall solve all grievances related to the course under consideration.

26. TRANSITORY REGULATIONS

A candidate, who is detained or has discontinued a semester, on readmission shall be required to do all the courses in the curriculum prescribed for the batch of students in which the student joins subsequently. However, exemption will be given to those candidates who have already passed such courses in the earlier semester(s) he was originally admitted into and substitute courses were offered in place of them as decided by the Board of Studies. However, the decision of the Board of Studies will be final.

a) Transfer candidates (from non-autonomous college affiliated to JNTUH):

A student who is following JNTUH curriculum, transferred from other college to this institute in third semester or subsequent semesters shall join with the autonomous batch in the appropriate semester. Such candidates shall be required to pass in all the courses in the program prescribed by the Board of Studies concerned for that batch of students from that semester onwards to be eligible for the award of degree. However, exemption will be given in the courses of the semester(s) of the batch which he had passed earlier and substitute courses are offered in their place as decided by the Board of Studies. The student has to clear all his backlog courses up to previous semester by appearing for the supplementary examinations conducted by JNTUH for the award of degree. The total number of credits to be secured for the award of the degree will be the sum of the credits up to the previous semester under JNTUH regulations and the credits prescribed for the semester in which a candidate joined after transfer and subsequent semesters under the autonomous status. The class will be awarded based on the academic performance of a student in the autonomous pattern.

b) Transfer candidates (from an autonomous college affiliated to JNTUH):

A student who has secured the required credits up to previous semester as per the regulations of other autonomous institutions shall also be permitted to be transferred to this institute. A student who is transferred from the other autonomous colleges to this institute in third semester or subsequent semesters shall join with the autonomous batch in the appropriate semester. Such candidates shall be required to pass in all the courses in the program prescribed by the Board of Studies concerned for that batch of students from that semester onwards to be eligible for the award of degree. However, exemption will be given in the courses of the semester(s) of the batch which he had passed earlier and substitute courses are offered in their place as decided by the Board of Studies. The total number of credits to be secured for the award of the degree will be the sum of the credits up to previous semester as per the regulations of the college from which he is transferred and the credits prescribed for the semester in which a candidate joined after transfer and subsequent semesters under the autonomous status. The class will be awarded based on the academic performance of a student in the autonomous pattern.

c) Readmission from R18 / UG20 to BT23 regulations

A student took admission in BT23 Regulations, detained due to lack of required number of credits or shortage of attendance at the end of any semester is permitted to take re-admission at appropriate level under any regulations prevailing in the institute course to the following rules and regulations.

1. Student shall pass all the courses in the earlier scheme of regulations. However, in case of having backlog courses, they shall be cleared by appearing for supplementary examinations conducted under earlier regulations from time to time.
2. After readmission, the student is required to study the courses as prescribed in the new regulations for the re-admitted program at that level and thereafter.
3. If the student has already passed any course(s) of readmitted program in the earlier regulation / semester of study, such courses are exempted.
4. The courses that are not done in the earlier regulations / semester as compared need to be cleared after readmission by appearing for the examinations conducted time to time under the new regulations.
5. In general, after transition, course composition and number of credits / semester shall be balanced between old and new regulations on case to case basis.
6. In case, the students who do not have option of acquiring required credits with the existing courses offered as per the new curriculum under autonomy, credit balance can be achieved by clearing the additional courses offered. The additional courses that are offered can be of theory or laboratory courses.

27. REVISION OF REGULATIONS AND CURRICULUM

The Institute from time to time may revise, amend or change the regulations, scheme of examinations and syllabi if found necessary and on approval by the Academic Council and the Governing Body shall be binding on the students, faculty, staff, all authorities of the Institute and others concerned.

FREQUENTLY ASKED QUESTIONS AND ANSWERS ABOUT AUTONOMY

1. Who grants Autonomy? UGC, Govt., AICTE or University

In case of Colleges affiliated to a university and where statutes for grant of autonomy are ready, it is the respective University that finally grants autonomy but only after concurrence from the respective state Government as well as UGC. The State Government has its own powers to grant autonomy directly to Govt. and Govt. aided Colleges.

2 Shall IARE award its own Degrees?

No. Degree will be awarded by Jawaharlal Nehru Technological University, Hyderabad with a mention of the name IARE on the Degree Certificate.

3 What is the difference between a Deemed University and an Autonomy College?

A Deemed University is fully autonomous to the extent of awarding its own Degree. A Deemed University is usually a Non-Affiliating version of a University and has similar responsibilities like any University. An Autonomous College enjoys Academic Autonomy alone. The University to which an autonomous college is affiliated will have checks on the performance of the autonomous college.

4 How will the Foreign Universities or other stake – holders know that we are an Autonomous College?

Autonomous status, once declared, shall be accepted by all the stake holders. The Govt. of Telangana mentions autonomous status during the First Year admission procedure. Foreign Universities and Indian Industries will know our status through our website.

5 What is the change of Status for Students and Teachers if we become Autonomous?

An autonomous college carries a prestigious image. Autonomy is actually earned out of our continued past efforts on academic performances, our capability of self- governance and the kind of quality education we offer.

6 Who will check whether the academic standard is maintained / improved after Autonomy? How will it be checked?

There is a built in mechanism in the autonomous working for this purpose. An Internal Committee called Academic Program Evaluation Committee, which will keep a watch on the academics and keep its reports and recommendations every year. In addition the highest academic council also supervises the academic matters. The standards of our question papers, the regularity of academic calendar, attendance of students, speed and transparency of result declaration and such other parameters are involved in this process.

7 Will the students of IARE as an Autonomous College qualify for University Medals and Prizes for academic excellence?

No. IARE has instituted its own awards, medals, etc. for the academic performance of the students. However for all other events like sports, cultural on co-curricular organized by the University the students shall qualify.

8 Can IARE have its own Convocation?

No. Since the University awards the degree the convocation will be that of the University, but there will be Graduation Day at IARE.

9 Can IARE give a provisional degree certificate?

Since the examinations are conducted by IARE and the results are also declared by IARE, the college sends a list of successful candidates with their final Grades and Grade Point Averages including CGPA to the affiliating university. Therefore with the prior permission of the university the institute will be entitled to give the provisional certificate.

10 Will Academic Autonomy make a positive impact on the Placements or Employability?

Certainly. The number of students qualifying for placement interviews is expected to improve, due to rigorous and repetitive classroom teaching and continuous assessment. Also the autonomous status is more responsive to the needs of the industry. As a result therefore, there will be a lot of scope for industry oriented skill development built-in into the system. The graduates from an autonomous college will therefore represent better employability.

11 What is the proportion of Internal and External Assessment as an Autonomous College?

Presently, it is 60% external and 40% internal. As the autonomy matures the internal assessment component shall be increased at the cost of external assessment.

12 Is it possible to have complete Internal Assessment for Theory or Practicals?

Yes indeed. We define our own system. We have the freedom to keep the proportion of external and internal assessment component to choose.

13 Why Credit based Grade System?

The credit based grade system is an accepted standard of academic performance the world over in all Universities. The acceptability of our graduates in the world market shall improve.

14 What exactly is a Credit based Grade System?

The credit based grade system defines a much better statistical way of judging the academic performance. One Lecture Hour per week of Teaching Learning process is assigned One Credit. One hour of laboratory work is assigned half credit. Letter Grades like A, B,C,D, etc. are assigned for a Range of Marks. (e.g. 91% and above is A+, 80 to 90 % could be A etc.) in Absolute Grading System while grades are awarded by statistical analysis in relative grading system. We thus dispense with sharp numerical boundaries. Secondly, the grades are associated with defined Grade Points in the scale of 1 to 10. Weighted Average of Grade Points is also defined Grade Points are weighted by Credits and averaged over total credits in a Semester. This process is repeated for all Semesters and a CGPA defines the Final Academic Performance

15 What are the norms for the number of Credits per Semester and total number of Credits for UG/PG program?

These norms are usually defined by UGC or AICTE. Usually around 25 Credits per semester is the accepted norm.

16 What is a Semester Grade Point Average (SGPA)?

The performance of a student in a semester is indicated by a number called SGPA. The SGPA is the weighted average of the grade points obtained in all the courses registered by the student during the semester.

$$SGPA = \sum_{i=1}^n (C_i G_i) / \sum_{i=1}^n C_i$$

Where, C_i is the number of credits of the i^{th} course and G_i is the grade point scored by the student in the i^{th} course and i represent the number of courses in which a student registered in the concerned semester. SGPA is rounded to two decimal places.

17 What is a Cumulative Grade Point Average (CGPA)?

An up-to-date assessment of overall performance of a student from the time of his first registration is obtained by calculating a number called CGPA, which is weighted average of the grade points obtained in all the courses registered by the students since he entered the Institute.

$$CGPA = \frac{\sum_{j=1}^m (C_j S_j)}{\sum_{j=1}^m C_j}$$

Where, S_j is the SGPA of the j^{th} semester and C_j is the total number of credits upto the semester and m represent the number of semesters completed in which a student registered upto the semester. CGPA is rounded to two decimal places.

18 Is there any Software available for calculating Grade point averages and converting the same into Grades?

Yes, The institute has its own MIS software for calculation of SGPA, CGPA, etc.

19 Will the teacher be required to do the job of calculating SGPAs etc. and convert the same into Grades?

No. The teacher has to give marks obtained out of whatever maximum marks as it is. Rest is all done by the computer.

20 Will there be any Revaluation or Re-Examination System?

No. There will double valuation of answer scripts. There will be a makeup Examination after a reasonable preparation time after the End Semester Examination for specific cases mentioned in the Rules and Regulations. In addition to this, there shall be a 'summer term' (compressed term) followed by the End Semester Exam, to save the precious time of students.

21 How fast Syllabi can be and should be changed?

Autonomy allows us the freedom to change the syllabi as often as we need.

22 Will the Degree be awarded on the basis of only final year performance?

No. The CGPA will reflect the average performance of all the semester taken together.

23 What are Statutory Academic Bodies?

Governing Body, Academic Council, Examination Committee and Board of Studies are the different statutory bodies. The participation of external members in everybody is compulsory. The institute has nominated professors from IIT, NIT, University (the officers of the rank of Pro-vice Chancellor, Deans and Controller of Examinations) and also the reputed industrialist and industry experts on these bodies.

24 Who takes Decisions on Academic matters?

The Governing Body of institute is the top academic body and is responsible for all the academic decisions. Many decisions are also taken at the lower level like Boards of Studies. Decisions taken at the Board of Studies level are to be ratified at the Academic Council and Governing Body.

25 What is the role of Examination committee?

The Examinations Committee is responsible for the smooth conduct of internal, End Semester and make up Examinations. All matters involving the conduct of examinations spot valuations, tabulations preparation of Grade Sheet etc fall within the duties of the Examination Committee.

26 Is there any mechanism for Grievance Redressal?

The institute has grievance redressal committee, headed by Dean - Student affairs and Dean - IQAC.

27 How many attempts are permitted for obtaining a Degree?

All such matters are defined in Rules & Regulation.

28 Who declares the result?

The result declaration process is also defined. After tabulation work wherein the SGPA, CGPA and final Grades are ready, the entire result is reviewed by the Moderation Committee. Any unusual

deviations or gross level discrepancies are deliberated and removed. The entire result is discussed in the Examinations and Result Committee for its approval. The result is then declared on the institute notice boards as well put on the web site and Students Corner. It is eventually sent to the University.

29 Who will keep the Student Academic Records, University or IARE?

It is the responsibility of the Examination Control Office of the institute to keep and preserve all the records.

30 What is our relationship with the JNT University?

We remain an affiliated college of the JNT University. The University has the right to nominate its members on the academic bodies of the college.

31 Shall we require University approval if we want to start any New Courses?

Yes, It is expected that approvals or such other matters from an autonomous college will receive priority.

32 Shall we get autonomy for PG and Doctoral Programs also?

Yes, presently our PG programs also enjoying autonomous status.

MALPRACTICE RULES

DISCIPLINARY ACTION FOR / IMPROPER CONDUCT IN EXAMINATIONS

S. No	Nature of Malpractices / Improper conduct	Punishment
	<i>If the candidate:</i>	
1. (a)	Possesses or keeps accessible in examination hall, any paper, note book, programmable calculator, cell phone, pager, palm computer or any other form of material concerned with or related to the course of the examination (theory or practical) in which he is appearing but has not made use of (material shall include any marks on the body of the candidate which can be used as an aid in the course of the examination)	Expulsion from the examination hall and cancellation of the performance in that course only.
(b)	Gives assistance or guidance or receives it from any other candidate orally or by any other body language methods or communicates through cell phones with any candidate or persons in or outside the exam hall in respect of any matter.	Expulsion from the examination hall and cancellation of the performance in that course only of all the candidates involved. In case of an outsider, he will be handed over to the police and a case is registered against him.
2.	Has copied in the examination hall from any paper, book, programmable calculators, palm computers or any other form of material relevant to the course of the examination (theory or practical) in which the candidate is appearing.	Expulsion from the examination hall and cancellation of the performance in that course and all other courses the candidate has already appeared including practical examinations and project work and shall not be permitted to appear for the remaining examinations of the courses of that Semester/year. The Hall Ticket of the candidate is to be cancelled and sent to the Controller of Examinations.
3.	Impersonates any other candidate in connection with the examination.	The candidate who has impersonated shall be expelled from examination hall. The candidate is also debarred and forfeits the seat. The performance of the original candidate, who has been impersonated, shall be cancelled in all the courses of the examination (including practicals and project work) already appeared and shall not be allowed to appear for examinations of the remaining courses of that semester/year. The candidate is also debarred for two consecutive semesters from class work and all semester end examinations. The continuation of the course by the candidate is course to the academic regulations in connection with forfeiture of seat. If the imposter is an outsider, he will be handed over to the police and a case is registered against him.
4.	Smuggles in the Answer book or additional sheet or takes out or arranges to send out the question paper during the examination or answer book or additional sheet, during or after the examination.	Expulsion from the examination hall and cancellation of performance in that course and all the other courses the candidate has already appeared including practical examinations and project work and shall not be permitted for the

		remaining examinations of the courses of that semester/year. The candidate is also debarred for two consecutive semesters from class work and all semester end examinations. The continuation of the course by the candidate is course to the academic regulations in connection with forfeiture of seat.
5.	Uses objectionable, abusive or offensive language in the answer paper or in letters to the examiners or writes to the examiner requesting him to award pass marks.	Cancellation of the performance in that course.
6.	Refuses to obey the orders of the Controller of Examinations /Additional Controller of Examinations/any officer on duty or misbehaves or creates disturbance of any kind in and around the examination hall or organizes a walk out or instigates others to walk out, or threatens the COE or any person on duty in or outside the examination hall of any injury to his person or to any of his relations whether by words, either spoken or written or by signs or by visible representation, assaults the COE or any person on duty in or outside the examination hall or any of his relations, or indulges in any other act of misconduct or mischief which result in damage to or destruction of property in the examination hall or any part of the Institute premises or engages in any other act which in the opinion of the officer on duty amounts to use of unfair means or misconduct or has the tendency to disrupt the orderly conduct of the examination.	In case of students of the college, they shall be expelled from examination halls and cancellation of their performance in that course and all other courses the candidate(s) has (have) already appeared and shall not be permitted to appear for the remaining examinations of the courses of that semester/year. The candidates also are debarred and forfeit their seats. In case of outsiders, they will be handed over to the police and a police case is registered against them.
7.	Leaves the exam hall taking away answer script or intentionally tears off the script or any part thereof inside or outside the examination hall.	Expulsion from the examination hall and cancellation of performance in that course and all the other courses the candidate has already appeared including practical examinations and project work and shall not be permitted for the remaining examinations of the courses of that semester/year. The candidate is also debarred for two consecutive semesters from class work and all semester end examinations. The continuation of the course by the candidate is course to the academic regulations in connection with forfeiture of seat.
8.	Possess any lethal weapon or firearm in the examination hall.	Expulsion from the examination hall and cancellation of the performance in that course and all other courses the candidate has already appeared including practical examinations and project work and shall not be permitted for the remaining examinations of the courses of that semester/year. The candidate is also debarred and forfeits the seat.
9.	If student of the college, who is not a candidate for the particular examination or any person not connected with the college indulges in any	Student of the colleges expulsion from the examination hall and cancellation of the performance in that course and all other

	malpractice or improper conduct mentioned in clause 6 to 8.	<p>courses the candidate has already appeared including practical examinations and project work and shall not be permitted for the remaining examinations of the courses of that semester/year. The candidate is also debarred and forfeits the seat.</p> <p>Person(s) who do not belong to the College will be handed over to police and, a police case will be registered against them.</p>
10.	Comes in a drunken condition to the examination hall.	Expulsion from the examination hall and cancellation of the performance in that course and all other courses the candidate has already appeared including practical examinations and project work and shall not be permitted for the remaining examinations of the courses of that semester/year.
11.	Copying detected on the basis of internal evidence, such as, during valuation or during special scrutiny.	Cancellation of the performance in that course and all other courses the candidate has appeared including practical examinations and project work of that semester/year examinations.
12.	If any malpractice is detected which is not covered in the above clauses 1 to 11 shall be reported to the University for further action to award suitable punishment.	

**FAILURE TO READ AND UNDERSTAND
THE REGULATIONS IS NOT AN EXCUSE**



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal, Hyderabad - 500 043

COURSE CATALOGUE

REGULATIONS: BT23

AERONAUTICAL ENGINEERING

I SEMESTER

Course Code	Course Name	Subject Area	Category	Periods Per Week			Credits	Scheme of Examination Max. Marks			
				L	T	P		CIA	SEE	Total	
INDUCTION PROGRAM				TWO WEEKS MANDATORY AUDIT COURSE							
THEORY											
AHSD01	Professional Communication	HSMC	Foundation	3	0	0	3	40	60	100	
AHSD02	Matrices and Calculus	BSC	Foundation	3	1	0	4	40	60	100	
AEED01	Elements of Electrical and Electronics Engineering	ESC	Foundation	3	0	0	3	40	60	100	
ACSD01	Object Oriented Programming	ESC	Foundation	3	0	0	3	40	60	100	
PRACTICAL											
AHSD04	Professional Communication Laboratory	HSMC	Foundation	0	0	2	1	40	60	100	
AEED03	Electrical and Electronics Engineering Laboratory	ESC	Foundation	0	0	2	1	40	60	100	
ACSD02	Object Oriented Programming with Java Laboratory	ESC	Foundation	0	1	2	2	40	60	100	
AMED01	Engineering Workshop	ESC	Foundation	1	0	2	2	40	60	100	
EXPERIENTIAL ENGINEERING EDUCATION (ExEED)											
ACSD03	Essentials of Innovation	Skill	Skill	0	0	2	1	40	60	100	
MANDATORY COURSE											
AHSD06	Environmental Science	MC	MC - I	Ref: Academic Regulations BT23							
TOTAL				13	02	10	20	360	540	900	

II SEMESTER

Course Code	Course Name	Subject Area	Category	Periods Per Week			Credits	Scheme of Examination Max. Marks		
				L	T	P		CIA	SEE	Total
THEORY										
AHSD03	Engineering Chemistry	BSC	Foundation	3	0	0	3	40	60	100
AHSD07	Applied Physics	BSC	Foundation	3	0	0	3	40	60	100
AHSD08	Differential Equations and Vector Calculus	BSC	Foundation	3	1	0	4	40	60	100
AMED04	Engineering Mechanics	ESC	Foundation	3	0	0	3	40	60	100
PRACTICAL										
AHSD05	Engineering Chemistry Laboratory	BSC	Foundation	0	0	2	1	40	60	100
AHSD09	Applied Physics Laboratory	BSC	Foundation	0	0	2	1	40	60	100
AMED05	Computer Aided Engineering Drawing	ESC	Foundation	1	0	2	2	40	60	100
ACSD06	Programming for Problem Solving Laboratory	ESC	Foundation	0	1	2	2	40	60	100
SKILL ENHANCEMENT PROJECT										
ACSD07	Mobile and Web Applications Development	Skill	Skill	0	0	2	1	40	60	100
MANDATORY COURSE										
AHSD10	Gender Sensitization	MC	MC - II	Ref: Academic Regulations BT-23						
FIELD PROJECT										
TOTAL				13	02	10	20	360	540	900

III SEMESTER

Course Code	Course Name	Subject Area	Category	Periods per week			Credits	Scheme of Examination Max. Marks		
				L	T	P		CIA	SEE	Total
THEORY										
AHSD11	Probability and Statistics	BSC	Foundation	3	1	0	4	40	60	100
AAED01	Mechanics of Solids	PCC	Core	3	0	0	3	40	60	100
AAED02	Thermodynamics and Heat Transfer	PCC	Core	3	0	0	3	40	60	100
AAED03	Fluid Dynamics	PCC	Core	3	0	0	3	40	60	100
ACSD08	Data Structures	ESC	Foundation	3	0	0	3	40	60	100
PRACTICAL										
AAED04	Numerical Methods using MATLAB	PCC	Core	0	0	2	1	40	60	100
AAED05	Mechanics of Solids & Fluid Dynamics Laboratory	PCC	Core	0	0	2	1	40	60	100
ACSD11	Data Structures Laboratory	ESC	Foundation	0	0	2	1	40	60	100
EXPERIENTIAL ENGINEERING EDUCATION (ExEED)										
ACSD12	Prototype and Design Building	Skill	Skill	0	0	2	1	40	60	100
TOTAL				15	01	08	20	360	540	900

IV SEMESTER

Course Code	Course Name	Subject Area	Category	Periods per week			Credits	Scheme of Examination Max. Marks		
				L	T	P		CIA	SEE	Total
THEORY										
AAED06	Aircraft Structures	PCC	Core	3	0	0	3	40	60	100
AAED07	Aircraft Propulsion and Turbomachinery	PCC	Core	3	0	0	3	40	60	100
AAED08	Aerodynamics	PCC	Core	3	0	0	3	40	60	100
AAED09	Flight Mechanics	PCC	Core	3	0	0	3	40	60	100
AAED10	Aerospace Materials and Production Technology	PCC	Core	3	0	0	3	40	60	100
PRACTICAL										
AAED11	Aerospace Structures Laboratory	PCC	Core	0	0	2	1	40	60	100
AAED12	Aerodynamics and Propulsion Laboratory	PCC	Core	0	0	2	1	40	60	100
AAED13	Aerospace Materials and Production Technology Laboratory	PCC	Core	0	0	2	1	40	60	100
SKILL ENHANCEMENT PROJECT										
ACSD18	DevOps Engineering	Skill	Skill	1	0	2	2	40	60	100
VALUE ADDED COURSE										
INTERNSHIP										
TOTAL				16	00	08	20	360	540	900

V SEMESTER

Course Code	Course Name	Subject Area	Category	Periods per week			Credits	Scheme of Examination Max. Marks		
				L	T	P		CIA	SEE	Total
THEORY										
AAED14	Artificial Intelligence for Aerospace Engineering	PCC	Core	3	0	0	3	40	60	100
AAED15	Aerospace Propulsion	PCC	Core	3	0	0	3	40	60	100
AAED16	Analysis of Aircraft Structures	PCC	Core	3	0	0	3	40	60	100
AAED17	Gas Dynamics	PCC	Core	3	0	0	3	40	60	100
	Program Elective – I	PEC	Elective	3	0	0	3	40	60	100
PRACTICAL										
AAED23	Aircraft Production Drawing Laboratory	PCC	Core	0	0	2	1	40	60	100
AAED24	Computer Aided Manufacturing Laboratory	PCC	Core	0	0	2	1	40	60	100
SKILL ENHANCEMENT PROJECT										
ACSD29	Engineering Design Project	Skill	Skill	0	0	2	1	40	60	100
ACSD30	Java Full Stack Development	Skill	Skill	1	0	2	2	40	60	100
VALUE ADDED COURSE										
TOTAL				16	00	08	20	360	540	900

#The course would consist of talks by working professionals from industry, government, academia & research organizations.

VI SEMESTER

Course Code	Course Name	Subject Area	Category	Periods per week			Credits	Scheme of Examination Max. Marks		
				L	T	P		CIA	SEE	Total
THEORY										
AAED25	Computational Aerodynamics and Turbulence Modeling	PCC	Core	3	0	0	3	40	60	100
AAED26	Finite Element Analysis	PCC	Core	3	0	0	3	40	60	100
	Program Elective – II	PEC	Elective	3	0	0	3	40	60	100
	Program Elective – III	PEC	Elective	3	0	0	3	40	60	100
	Open Elective – I	OEC	Elective	3	0	0	3	40	60	100
PRACTICAL										
AAED39	Computational Aerodynamics and Turbulence Modeling Laboratory	PCC	Core	0	0	2	1	40	60	100
AAED40	Computational Structure Laboratory	PCC	Core	0	0	2	1	40	60	100
SKILL ENHANCEMENT PROJECT										
ACSD44	Data Scientist / AI Specialist	Skill	Skill	1	0	2	2	40	60	100
ACSD45	Development Project	Skill	Skill	0	0	2	1	40	60	100
VALUE ADDED COURSE										
INTERNSHIP										
TOTAL				16	00	08	20	360	540	900

VII SEMESTER

Course Code	Course Name	Subject Area	Category	Periods per week			Credits	Scheme of Examination Max. Marks		
				L	T	P		CIA	SEE	Total
THEORY										
AAED41	Flight Vehicle Design	PCC	Core	3	0	0	3	40	60	100
AAED42	Aerospace Structural Dynamics	PCC	Core	3	0	0	3	40	60	100
	Program Elective – IV	PEC	Elective	3	0	0	3	40	60	100
	Program Elective – V	PEC	Elective	3	0	0	3	40	60	100
	Open Elective – II	OEC	Elective	3	0	0	3	40	60	100
PRACTICAL										
AAED55	Flight Vehicle Design Laboratory	PCC	Core	0	0	2	1	40	60	100
AAED56	Aerospace Structural Dynamics Laboratory	PCC	Core	0	0	2	1	40	60	100
PROJECT WORK										
AAED57	Project Work (Phase - I)	PROJ	Project	0	0	6	3	40	60	100
MANDATORY COURSE										
AHSD14	Essence of Indian Traditional Knowledge	MC-2	MC	Ref: Academic Regulations BT-23						
TOTAL				15	00	10	20	320	480	800

VIII SEMESTER

Course Code	Course Name	Subject Area	Category	Periods per week			Credits	Scheme of Examination Max. Marks		
				L	T	P		CIA	SEE	Total
THEORY										
AHSD15	Managerial Economics and Financial Analysis	BSC	Foundation	3	0	0	3	40	60	100
	Program Elective - VI	PEC	Elective	3	0	0	3	40	60	100
	Open Elective-III	OEC	Elective	3	0	0	3	40	60	100
PROJECT WORK										
AAED65	Project Work (Phase - II)	PCC	Project	0	0	22	11	40	60	100
TOTAL				09	00	22	20	160	240	400

ELECTIVE COURSES

PROGRAM ELECTIVES COURSES (PEC)

Course Code	Name of the Course	Prerequisites	Preferred Semester	Credits
AAED18	Heat and Mass Transfer	Thermodynamics	V	3
AAED19	Mechanism and Machine Design	Mechanics of Solids	V	3
AAED20	CAE/CAM	Aerospace Materials and Production Technology	V	3
AAED21	Aircraft Systems and Control	Introduction to Aerospace Engineering	V	3
AAED22	Space Dynamics	Aerodynamics	V	3
AAED27	Aircraft Systems and Instrumentation	Aircraft Performance	VI	3
AAED28	Rocket and Missile Technology	Aerospace Propulsion	VI	3
AAED29	Techniques in Wind Tunnel Testing	Fluid Dynamics	VI	3
AAED30	Fatigue and Fracture of Materials	Aircraft Structures	VI	3
AAED31	Orbital Mechanics	Aircraft Propulsion	VI	3
AAED32	Turbo Machinery	Fluid Dynamics	VI	3
AAED33	Theory of Stress Strain Measurements	Aircraft Structures	VI	3
AAED34	High Temperature Gas Dynamics	Gas Dynamics	VI	3
AAED35	Gas Turbines and Jet Propulsion Technology	Aircraft Propulsion	VI	3
AAED36	Unmanned Air Vehicles	Fluid Dynamics	VI	3
AAED43	Computational Fluid Dynamics	Fluid Dynamics	VII	3
AAED44	Introduction to Composite Materials	Aerospace Materials and Production Technology	VII	3
AAED45	Air Transportation System	Introduction to Aerospace Engineering	VII	3
AAED46	Hypersonic Aerodynamics	Gas Dynamics	VII	3
AAED47	Ground Vehicle Aerodynamics	Aerodynamics	VII	3
AAED48	Theory of Aeroelasticity	Aircraft Structures	VII	3
AAED49	Avionics and Instrumentation	Elements of Electrical and Electronics Engineering	VII	3
AAED50	Automatic Control of Aircraft	Aircraft Stability and Control	VII	3
AAED51	Helicopter Aerodynamics	Aerodynamics	VII	3
AAED52	Introduction to Nano Technology	Aerospace Materials and Production Technology	VII	3
AAED58	Flight Scheduling and Operations	Introduction to Aerospace Engineering	VIII	3
AAED59	Non Destructive Testing	Aerospace Materials and Production Technology	VIII	3
AAED60	Engineering Optimization Techniques	Probability and Statistics	VIII	3
AAED61	Molecular Gas Dynamics	Gas Dynamics	VIII	3
AAED62	Aircraft Maintenance Engineering	Introduction to Aerospace Engineering	VIII	3

OPEN ELECTIVE COURSES (OEC)

The courses listed below are offered by the department of Aeronautical Engineering for students of other departments.

Course Code	Course Name	Credits
AAED37	Elements of Aerospace Engineering	3
AAED38	Drone Technology	3
AAED53	Vehicle Aerodynamics	3
AAED54	Building Aerodynamics	3
AAED63	Space Engineering	3
AAED64	Jet Propulsion Systems	3

VALUE-ADDED COURSES

Objective:

Value added courses are provided to equip the students with knowledge and skills outside of the curriculum or to meet any specific requirements of the industry. The following are the Value-Added Courses provided to students by various departments in our institution.

ANY THREE FROM THE GIVEN BELOW THROUGH IARE ELRV – AKANKSHA / CERTIFICATE - SWAYAM, e-PG pathshala, NPTEL etc..

1. Data Scalability and Distribution (Amazon Web Services, Microsoft Azure, Google Cloud Platform etc.)
2. Software Developer (Restful webservices / Microservices, Rust programming, MEAN, MERN, MEVN)
3. Data Science (Data visualization, Data wrangling, Bigdata Technologies, Business Intelligence etc..)
4. Operating Systems (IBM I, Mac OS, Linux, Haiku etc.)
5. Debugging
6. Testing (Selenium, TestNG)
7. Cyber Security (Network Security, Threat Intelligence and Analysis / Risk Assessment and Management)
8. Software Architect
9. Blockchain Technology



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

PROFESSIONAL COMMUNICATION								
I Semester: AE / ME / CE / CSE (AI &ML) / IT / ECE / EEE								
II Semester: CSE CSE (DS) CSE (CS)								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AHSD01	Foundation	L	T	P	C	CIA	SEE	Total
		3	-	-	3	40	60	100
Contact Classes: 64	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: 48			
Prerequisite:								

I. COURSE OVERVIEW:

The principle aim of the course is that the students will have awareness about the importance of English language in the contemporary times and also it emphasizes the students to learn this language as a skill (listening skill, speaking skill, reading skill and writing skill). Moreover, the course benefits the students how to solve their day-to-day problems in speaking English language. Besides, it assists the students to reduce the mother tongue influence and acquire the knowledge of neutral accent. The course provides theoretical and practical knowledge of English language and it enables students to participate in debates about informative, persuasive, didactic, and commercial purposes.

II. COURSE OBJECTIVES:

The students will try to learn:

- I Standard pronunciation, appropriate word stress, and necessary intonation patterns for effective communication towards achieving academic and professional targets.
- II Appropriate grammatical structures and also using the nuances of punctuation tools for practical purposes.
- III Critical aspect of speaking and reading for interpreting in-depth meaning between the sentences.
- IV Conceptual awareness on writing in terms of unity, content, coherence, and linguistic accuracy.

III. COURSE OUTCOMES:

After successful completion of the course, students should be able to:

- CO 1 Demonstrate the prime necessities of listening skills and communication skills for academic and non-academic purposes.
- CO 2 Explain effectively in spoken English on issues and ideas with a reasonable degree of fluency and accuracy in different social settings and different kinds of social encounters.
- CO 3 Strengthen acceptable language for developing life skills to overcome the challenges at professional platform.
- CO 4 Interpret the grammatical and lexical forms of English and use of these forms in specific communicative contexts.
- CO 5 Articulate main ideas, both stated and inferred, and important details in academic, journalistic, and literary prose at advanced level.
- CO 6 Extend writing skills for fulfilling academic and work-place requirements of various written communicative functions.

IV. COURSE CONTENT:

MODULE – I: GENERAL INTRODUCTION AND LISTENING SKILLS (13)

Introduction to communication skills, communication process, elements of communication, soft skills and hard skills, importance of soft skills for engineers, listening skills, significance of listening skills, stages of listening, barriers and effectiveness of listening, listening comprehension.

MODULE – II: SPEAKING SKILL (13)

Significance of speaking skills, essentials of speaking skills, verbal and non-verbal communication, generating

talks based on visual prompts, public speaking, exposure to structured talks, delivering speech effectively, oral presentation using power point slides.

MODULE – III: VOCABULARY AND GRAMMAR (13)

The concept of word formation, idioms and phrases, one-word substitutes, sentence structure (simple, compound and complex), usage of punctuation marks, advanced level prepositions.

Tenses, subject verb agreement, degrees of comparison, direct and indirect speech, questions tags.

MODULE – IV: READING SKILL (12)

Significance of reading skills, techniques of reading, skimming-reading for the gist of a text, scanning-reading for specific information, intensive, extensive reading, reading comprehension, metaphor and figurative language.

MODULE – V: WRITING SKILL (13)

Significance of writing skills, effectiveness of writing, the role of a topic sentence and supporting sentences in a paragraph, organizing principles of paragraphs in a document, writing introduction and conclusion, techniques for writing precis, various formats for letter writing (block format, full block format, and semi bloc format), e-mail writing, report writing.

V. TEXT BOOKS:

1. Anjana Tiwari, *Communication Skills in English*, Khanna Publishing House: New Delhi, 2022.

VI. REFERENCE BOOKS:

1. Norman Whitby, *Business Benchmark: Pre-Intermediate to Intermediate – BEC Preliminary*, Cambridge University Press, 2nd Edition, 2008.
2. Devaki Reddy, Shreesh Chaudhary, *Technical English*, Macmillan, 1st Edition, 2009.
3. Rutherford, Andrea J, *Basic Communication Skills for Technology*, Pearson Education, 2nd Edition, 2010.
4. Raymond Murphy, *Essential English Grammar with Answers*, Cambridge University Press, 2nd Edition, 2010.

VII. ELECTRONICS RESOURCES:

1. https://akanksha.iare.ac.in/index?route=course/details&course_id=954
2. https://akanksha.iare.ac.in/index?route=course/details&course_id=10
3. https://akanksha.iare.ac.in/index?route=course/details&course_id=352
4. <https://akanksha.iareac.in/index?route=publicprofile&id=5075>

VIII. MATERIALS ONLINE

1. Course template
2. Tech-talk topics
3. Assignments
4. Definition and terminology
5. Tutorial question bank
6. Model question paper – I
7. Model question paper – II
8. Lecture notes
9. Early lecture readiness videos (ELRV)
10. Power point presentations

COURSE CONTENT

MATRICES AND CALCULUS								
I Semester: AE / ME / CE / ECE / EEE / CSE / CSE (AI & ML) / CSE (DS) / CSE (CS) / IT								
Course Code	Category	Hours/Week			Credits	Maximum Marks		
AHSD02	Foundation	L	T	P	C	CIA	SEE	Total
		3	1	-	4	40	60	100
Contact Classes: 48	Tutorial Classes: 16	Practical Classes: Nil			Total Classes: 64			
Prerequisite: Basic Principles of Algebra and Calculus								

I. COURSE OVERVIEW:

This course Matrices and Calculus is a foundation course of mathematics for all engineering branches. The concepts of Matrices, Eigen Values, Eigen Vectors, Functions of Single and Several Variables, Fourier Series and Multiple Integrals. This course is applicable for simulations, colour imaging process, finding optimal solutions in all fields of industries.

II. COURSE OBJECTIVES:

The students will try to learn:

- I The concept of the rank of a matrix, solve the system of linear equations, eigen values, eigen vectors.
- II The geometrical approach to the mean value theorems and their application to the mathematical problems.
- III The Fourier series expansion in standard intervals as well as arbitrary intervals.
- IV The evaluation of multiple integrals and their applications.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO 1 Determine the rank and solutions of linear equations with elementary operations.
- CO 2 Utilize the Eigen values, Eigen vectors for developing spectral matrices.
- CO 3 Make use of Cayley-Hamilton theorem for finding powers of the matrix.
- CO 4 Interpret the maxima and minima of given functions by finding the partial derivatives.
- CO 5 Apply the Fourier series expansion of periodic functions for analyzing the wave forms.
- CO 6 Determine the area of solid bounded regions by using the integral calculus.

IV. COURSE CONTENT:

MODULE - I: MATRICES (09)

Rank of a matrix by echelon form and normal form, inverse of non-singular matrices by Gauss-Jordan method, system of linear equations, solving system of homogeneous and non-homogeneous equations.

MODULE - II: EIGEN VALUES AND EIGEN VECTORS (10)

Eigen values, eigen vectors and their properties (without proof), Cayley-Hamilton theorem (without proof), verification, finding inverse and power of a matrix by Cayley-Hamilton theorem, diagonalization of a matrix.

MODULE - III: FUNCTIONS OF SINGLE and SEVERAL VARIABLES (10)

Mean value theorems Rolle's theorem, Lagrange's theorem, Cauchy's theorem-without proof.

Functions of several variables: Partial differentiation, Jacobian, functional dependence, maxima and minima of functions of two variables and three variables, method of Lagrange multipliers.

MODULE – IV: FOURIER SERIES (09)

Fourier expansion of periodic function in a given interval of length 2π , Fourier series of even and odd functions, Fourier series in an arbitrary interval, half- range Fourier sine and cosine expansions.

MODULE – V: MULTIPLE INTEGRALS (10)

Evaluation of double integrals (cartesian and polar coordinates), change of order of integration (only cartesian coordinates), evaluation of triple integrals (only cartesian coordinates).

V. TEXT BOOKS:

1. B. S. Grewal, *Higher Engineering Mathematics*, 44/e, Khanna Publishers, 2017.
2. Erwin Kreyszig, *Advanced Engineering Mathematics*, 10/e, John Wiley & Sons, 2011.

VI. REFERENCE BOOKS:

1. R. K. Jain and S. R. K. Iyengar, *Advanced Engineering Mathematics*, 3/ed, Narosa Publications, 5th Edition, 2016.
2. George B. Thomas, Maurice D. Weir and Joel Hass, Thomas, *Calculus*, 13/e, Pearson Publishers, 2013.
3. N.P. Bali and Manish Goyal, *A text book of Engineering Mathematics*, Laxmi Publications, Reprint, 2008.
4. Dean G. Duffy, *Advanced Engineering Mathematics with MATLAB*, CRC Press.
5. Peter O’Neil, *Advanced Engineering Mathematics*, Cengage Learning.
6. B.V. Ramana, *Higher Engineering Mathematics*, McGraw Hill Education.

VII. ELECTRONICS RESOURCES:

1. Engineering Mathematics - I, By Prof. Jitendra Kumar | IIT Kharagpur
https://onlinecourses.nptel.ac.in/noc23_ma88/preview
2. Advanced Calculus for Engineers, By Prof. Jitendra Kumar, Prof. Somesh Kumar | IIT Kharagpur
https://onlinecourses.nptel.ac.in/noc23_ma86/preview
3. http://www.efunda.com/math/math_home/math.cfm
4. <http://www.ocw.mit.edu/resources/#Mathematics>
5. <http://www.sosmath.com>
6. <http://www.mathworld.wolfram.com>

VIII. MATERIAL ONLINE:

1. Course template
2. Tech-talk topics
3. Assignments
4. Definition and terminology
5. Tutorial question bank
6. Model question paper – I
7. Model question paper – II
8. Lecture notes
9. Early lecture readiness videos (ELRV)
10. Power point presentations



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

ELEMENTS OF ELECTRICAL AND ELECTRONICS ENGINEERING								
I Semester: CSE (AI&ML) / IT / AERO / MECH / CIVIL								
II Semester: CSE / CSE (DS) / CSE (CS)								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AEED01	Foundation	L	T	P	C	CIA	SEE	Total
		3	-	-	3	40	60	100
Contact Classes: 48	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: 48			
Prerequisite: Physics								

I. COURSE OVERVIEW:

This course enables knowledge on electrical quantities such as current, voltage, and power, energy to know the impact of technology in global and societal context. It provides the knowledge on basic DC and AC circuits used in electrical and electronic devices, highlights the importance of electrical machines and basics of semiconductor devices like diodes and transistors.

II. COURSES OBJECTIVES:

The students will try to learn

- I The fundamentals of electrical circuits and analysis of circuits with DC and AC excitation using circuit laws.
- II The construction and operation of Electrical machines.
- III The operational characteristics of semiconductor devices with their applications.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO 1 Make use of basic electrical laws for solving DC and AC circuits.
- CO 2 Apply network theorems for analysis of simple electrical circuits.
- CO 3 Demonstrate the fundamentals of electromagnetism for the operation of DC and AC machines.
- CO 4 Utilize the characteristics of semiconductor devices for the application of rectifiers and regulators.
- CO 5 Interpret the transistor configurations for optimization of the operating point.
- CO 6 Understand the amplifier circuits using transistors for calculating different parameters

IV. COURSE CONTENT:

MODULE-I: INTRODUCTION TO ELECTRICAL CIRCUITS (09)

Circuit concept: Ohm's law, Kirchhoff's laws, the equivalent resistance of networks, star to delta transformation, mesh and nodal analysis (with DC source only).

Single phase AC circuits: representation of alternating quantities, RMS, average, form and peak factor, RLC series circuit.

MODULE-II: NETWORK THEOREMS AND THREE-PHASE VOLTAGES (10)

Network Theorems: Superposition, reciprocity, Thevenin's, Norton's, Maximum power transfer theorems for DC excitation circuits, three phase voltages (Definitions only): Voltage and current relationships in star and delta connections.

MODULE-III: ELECTRICAL MACHINES AND SEMICONDUCTOR DIODES (10)

DC and AC machines: Motors and generators, principle of operation, parts, EMF equation, types, applications, losses and efficiency.

Semiconductor diode: P-N Junction diode, symbol, V-I characteristics, half wave rectifier, full wave rectifier, bridge rectifier and filters, diode as a switch, Zener diode as a voltage regulator.

MODULE-IV: BIPOLAR JUNCTION TRANSISTOR AND APPLICATIONS (10)

Bipolar junction transistor: Characteristics and configurations, working principle NPN and PNP transistor, CE, CB, CC configurations – input and output characteristics, transistor as a switch.

MODULE-V: TRANSISTOR AMPLIFIERS (09)

Amplifier circuits: Two port devices and network, small signal models for transistors, concept of small signal operation, amplification in CE amplifier, h parameter model of a BJT- CE, CB and emitter follower analysis.

V. TEXT BOOKS:

1. M. S. Sukhija, T. K. Nagsarkar, *Basic Electrical and Electronics Engineering*, Oxford, 1st Edition, 2012.
2. Salivahanan, *Electronics Devices & Circuits*, TMH 4th Edition 2012.

VI. REFERENCE BOOKS:

1. CL Wadhwa, *Electrical Circuit Analysis including Passive Network Synthesis*, International, 2nd Edition, 2009.
2. David A Bell, *Electric circuits*, Oxford University Press, 7th Edition, 2009.
3. PS Bimbira, *Electrical Machines*, Khanna Publishers, 2nd Edition, 2008.
4. D.P. Kothari and I. J. Nagrath, *Basic Electrical Engineering*, Tata McGraw Hill, 4th Edition, 2019.

VII. ELECTRONICS RESOURCES:

1. <https://www.kuet.ac.bd/webportal/ppmv2/uploads/1364120248DC%20Machines>
2. <https://www.eleccompengineering.files.wordpress.com/2014/08/a-textbook-of-electrical-technologyvolume-ii-ac-and-dc-machines-b-l-thferaja.pdf>
3. https://www.geosci.uchicago.edu/~moyer/GEOS24705/Readings/Klempner_Ch1.pdf
4. <https://www.ibiblio.org/kuphaldt/electricCircuits/DC/DC.pdf>
5. <https://www.users.ece.cmu.edu/~dwg/personal/sample.pdf>
6. <https://www.iare.ac.in>

VIII. MATERIALS ONLINE

1. Course template
2. Tutorial question bank
3. Tech talk topics
4. Definitions and terminology
5. Open ended experiments
6. Model question paper-i
7. Model question paper-ii
8. Lecture notes
9. Early learning readiness videos (elrv)
10. Power point presentations

COURSE CONTENT

OBJECT ORIENTED PROGRAMMING								
I Semester: AE / ME / CE / ECE / EEE / CSE / CSE (AI & ML) / CSE (DS) / CSE (CS) / IT								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
ACSD01	Foundation	L	T	P	C	CIA	SEE	Total
		3	0	0	3	40	60	100
Contact Classes: 48	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: 48			
Prerequisites: There are no prerequisites to take this course.								

I. COURSE OVERVIEW:

The course provides a solid foundation in object-oriented programming concepts in using them. It includes concepts object-oriented concepts such as information hiding, encapsulation, and polymorphism. It contrasts the use of inheritance and composition as techniques for software reuse. It provides an understanding of object-oriented design using graphical design notations such as Unified Modeling Language (UML) as well as object design patterns.

II. COURSES OBJECTIVES:

The students will try to learn

- The fundamental concepts and principles of object-oriented programming in high-level programming languages.
- The advanced concepts for developing well-structured and efficient programs that involve complex data structures, numerical computations, or domain-specific operations.
- The design and implementation of features such as inheritance, polymorphism, and encapsulation for tackling complex problems and creating well-organized, modular, and maintainable code.
- The usage of input/output interfaces to transmit and receive data to solve real-time computing problems.

III. COURSE OUTCOMES:

At the end of the course, students should be able to:

- CO 1 Interpret the features of object-oriented programming languages, comparison, and evaluation of programming languages.
- CO 2 Model the real-world scenario using class diagrams and exhibit communication between objects.
- CO 3 Estimate the need for special functions for data initialization.
- CO 4 Outline the features of object-oriented programming for binding the attributes and behavior of a real-world entity.
- CO 5 Use the concepts of streams and files that enable data management to enhance programming skills.
- CO 6 Develop contemporary solutions to software design problems using object-oriented principles.

IV. COURSE CONTENT:

MODULE - I: Object-oriented concepts (09)

Objects and legacy systems, procedural versus Object-oriented programming, top-down and bottom-up approaches and their differences, benefits of OOP, applications of OOP, features of OOP.

Abstraction: Layers of abstraction, forms of abstraction, abstraction mechanisms.

MODULE - II: Classes and objects (09)

Classes and objects: Object data, object behaviors, creating objects, attributes, methods, messages, creating class diagrams.

Access specifiers and initialization of class members: Accessing members and methods, access specifiers - public, private, protected, memory allocation. Static members, static methods.

MODULE - III: Special member functions and overloading (09)

Constructors and destructors: Need for constructors and destructors, copy constructors, dynamic constructors, parameterized constructors, destructors, constructors and destructors with static members.

Overloading: Function overloading, constructor overloading, and operator overloading - rules for overloading operators, overloading unary and binary operators, friend functions.

MODULE – IV: Inheritance and polymorphism (09)

Inheritance: types of inheritance, base class, derived class, usage of final, ambiguity in multiple and multipath inheritance, virtual base class, overriding member functions, order of execution of constructors and destructors.

Polymorphism and virtual functions: Virtual functions, pure virtual functions, abstract classes, introduction to polymorphism, static polymorphism, dynamic polymorphism.

MODULE –V: Console I/O and working with files (09)

Console I/O: Concept of streams, hierarchy of console stream classes, unformatted I/O operations, managing output with manipulators.

Working with files: Opening, reading, writing, appending, processing, and closing different types of files, command line arguments.

V. TEXTBOOKS:

1. Matt Weisfeld, *The Object-Oriented Thought Process*, Addison Wesley Object Technology Series, 4th Edition, 2013.

VI. REFERENCE BOOKS:

1. Timothy Budd, *Introduction to object-oriented programming*, Addison Wesley Object Technology Series, 3rd Edition, 2002.
2. Gaston C. Hillar, *Learning Object-Oriented Programming*, Packt Publishing, 2015.
3. Kingsley Sage, *Concise Guide to Object-Oriented Programming*, Springer International Publishing, 1st Edition, 2019.
4. Rudolf Pecinovsky, *OOP - Learn Object Oriented Thinking and Programming*, Tomas Bruckner, 2013.
5. Grady Booch, *Object-oriented analysis and design with applications*, Addison Wesley Object Technology Series, 3rd Edition, 2007.

VII. ELECTRONICS RESOURCES:

1. <https://docs.oracle.com/javase/tutorial/java/concepts/>
2. <https://www.w3schools.com/cpp/>
3. <https://www.edx.org/learn/object-oriented-programming/>
4. <https://www.geeksforgeeks.org/introduction-of-object-oriented-programming/>

VIII. MATERIALS ONLINE

1. Course template
2. Tutorial question bank
3. Tech talk topics
4. Open-ended experiments
5. Definitions and terminology
6. Assignments
7. Model question paper – I
8. Model question paper – II
9. Lecture notes
10. PowerPoint presentation
11. E-Learning Readiness Videos (ELRV)



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

PROFESSIONAL COMMUNICATION LABORATORY								
I Semester: CSE (AI&ML) IT AE ECE EEE ME CE								
II Semester: CSE CSE(DS) CSE(CS)								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AHSD04	Foundation	L	T	P	C	CIA	SEE	Total
		-	-	2	1	40	60	100
Contact Classes: Nil	Tutorial Classes: Nil	Practical Classes: 45			Total Classes: 45			
Prerequisite: Nil								

I. COURSE OVERVIEW:

This laboratory course is designed to introduce students to create a wide exposure on language learning techniques of the basic elements of listening skills, speaking skills, reading skills and writing skills. In this laboratory, students are trained in communicative English language skills, phonetics, word accent, word stress, rhythm, intonation, oral presentations and extempore speeches. Students are also taught in terms of seminars, group-discussions, presenting techniques of writing, participating in role plays, telephonic etiquettes, asking and giving directions, information transfer, debates, description of persons, places and objects etc. The laboratory encourages students to work in a group, engage in peer-reviews and inculcate team spirit through various exercises on grammar, vocabulary, and pronunciation games etc. Students will make use of all these language skills in academic, professional and real time situations.

II. COURSES OBJECTIVES

The students will try to learn:

- I. English speech sounds, word accent, intonation and stress patterns for effective pronunciation.
- II. Critical aspect of speaking and reading for interpreting in-depth meaning between the sentences.
- III. Language techniques for social interactions such as public speaking, group discussions and interviews.
- IV. Computer-assisted multi-media instructions and independent language learning.

III. COURSE OUTCOMES:

At the end of the course, students will be able to:

- CO 1 Articulate acceptable accent in order to execute formal and informal communication.
- CO 2 Differentiate stress shifts, syllabification and make use of past tense and plural markers effectively in connected speech; besides participate in role plays with confidence.
- CO 3 Use weak forms and strong forms in spoken language and maintain intonation patterns as a native speaker to avoid mother tongue influence; moreover, practice various etiquettes at professional platform.
- CO 4 Demonstrate errors in pronunciation and the decorum of oral presentations; for that reason, take part joining in group discussions and debates with much critical observations.
- CO 5 Strengthen writing effective messages, notices, summaries and also able to write reviews very critically of art and academical videos.
- CO 6 Argue scholarly, giving the counters to open ended experiments, and also writing slogans for the products talentedly.

Dos

1. Turn up in a neat and formal dress code regularly and maintain punctuality.
2. Bring observation books and worksheets for every laboratory session without fail.
3. Keep lab record book up to date.
4. Students must adhere to the acceptable use of ICT resources policy.
5. CD ROM 's, USB and other multimedia equipment are for college use only.
6. Replace headsets onto the monitor and rearrange your chairs back into their position as you leave laboratory.
7. Get your lab worksheets evaluated and upload online within the stipulated time for online evaluation by the faculty concerned.
8. Conduct yourself at the best to be a good learner.

Don'ts

1. Do not use the on/off switch to reboot the system.
2. Do not breach copyright regulations.
3. Do not install or download any software or modify or delete any system files on any laboratory computer.
4. Do not read or modify other users' files.
5. Do not damage, remove, or disconnect any labels, parts, cables or equipment.
6. Do not make undue noise in the laboratories. Be considerate of the other lab users- this is study area.
7. No food and the beverages are allowed into computer laboratories.

IV. COURSE CONTENT

Exercises for professional communication laboratory

Note: Students are encouraged to bring their own laptops for laboratory practice session

Getting started exercises

1. Introduction to pronunciation

Experiments

CALL LAB: Introduction to pronunciation

ICS LAB: Introducing self and introducing others and feedback

Activities

- a) Pronunciation practice among groups.
- b) Follow formal rules of introducing self and others and giving the constructive feedback.
- c) Pronunciation practice from the K-Van software

2. Introduction to phonetics

Experiments

CALL LAB: Introduction to phonetics, listening to English sounds, Vowel and Consonant sounds.

ICS LAB: Describing a person or place or a thing using relevant adjectives – feedback

Activities

- a) Introduction to phonetics, listening to English sounds, Vowel and Consonant sounds
- b) Describing a person or place or a thing using relevant adjectives and giving valid feedback.
- c) Listening English phonemes from K-Van software

3. Syllabification of different lexemes

Experiments

CALL LAB: Structure of syllables.

ICS LAB: JAM Sessions using public address system

Activities

- a) Participating in just a minute session
- b) Practicing consonant clusters
- c) Practicing different methods of dividing the syllables

4. Word accent and stress shift patterns

Experiments:

CALL LAB: Word accent and stress shifts.

ICS LAB: Asking for directions and giving directions

Activities:

- a) Usage of appropriate lexical tools at the time of giving directions
- b) Pronounce weak and strong forms using strategies in spoken language

5. Usage of markers in pronunciation

Experiments:

CALL LAB: Past tense and plural markers.

ICS LAB: Role plays on fixed expressions in various situations

Activities:

- a) Addition of suffixes to verbs

- b) Multiple exercises on nouns based its number
- c) Execution of body language in different contexts

6. Use of form in connected speech

CALL LAB: Weak forms and strong forms

ICS LAB: Extempore-Picture

Activities:

- a) Practice on recognition of objects
- b) Preparation on connected speech based on tone boundaries
- c) Execution while describing pictures

7. Intonation patterns

Experiments

CALL LAB: Intonation

ICS LAB: Interpretation of Proverbs and Idioms

Activities

- a) Voce modulation in speech as per context
- b) Application of proverbs in real life contexts
- c) Exercises on idiomatic expressions

8. Neutralization of Mother Tongue influence (MTI)

Experiments

CALL LAB: Neutralization of Mother Tongue Influence (MTI).

ICS LAB: Etiquette

Activities

- a) Mirror practicing to get rid of mother tongue influence
- b) Listening to English speeches from native speakers
- c) Etiquettes for social and professional behavior

9. Oral presentations

Experiments

CALL LAB: Common errors in pronunciation practice through tongue twisters.

ICS LAB: Oral Presentations

Activities

- a) Practicing tongue twisters to accelerate language fluency
- b) Practice on how to grab attention of audience
- c) Formulating strategies for structured presentation

10. Minimal pairs and debates

Experiments

CALL LAB: Minimal pairs.

ICS LAB: Debates

Activities

- a) Practicing minimal pair dominoes
- b) Working on minimal pair-cards and counters
- c) Executing interpersonal skills with argumentative issues

11. Listening comprehension skills

Experiments

CALL LAB: Listening comprehension.

ICS LAB: Group discussion

Activities

- a) Listening to dialogues of native speakers
- b) Listening and writing short stories

- c) Exercising group discussion on various social issues

12. Enriching writing skills

Experiments

CALL LAB: Demonstration on how to write leaflets, messages and notices.

ICS LAB: Techniques and methods to write summaries and reviews of videos

Activities

- a) Practice on writing suitable messages and notices
- b) Summary writing techniques
- d) Practice on writing reviews for sociopolitical videos

13. Activities on pronunciation skills

Experiments

CALL LAB: Pronunciation practice.

ICS LAB: Information transfer

Activities

- a) Exercise on commonly mispronounced words
- b) Preparation on similar words pronunciation
- c) Practice on transferring information through chats and diagrams

14. Writing reviews after watching statements

Experiments

CALL LAB: Open ended experiments-phonetics practice.

ICS LAB: Providing reviews and remarks

Activities

- a) Writing three term labels for vowels
- b) Writing three term labels for consonants
- c) Practice on giving reviews and remarks

15. Experiments on text to speech and writing slogans

Experiments

CALL LAB: Open-ended experiments-text to speech.

ICS LAB: Writing slogan related to the image

Activities

- a) Practice through streaming video lessons
- b) Training on creating dialogues and stories
- c) Practice on designing and writing slogans

V. TEXT BOOKS:

1. Professional Communication laboratory manual

VI. REFERENCE BOOK

1. Meenakshi Raman, Sangeetha Sharma, *Technical Communication Principles and Practices*, Oxford University Press, New Delhi, 3rd Edition, 2015.
2. Rhirdion, Daniel, *Technical Communication*, Cengage Learning, New Delhi, 1st Edition, 2009.

VII. ELECTRONICS RESOURCES

1. Cambridge online pronunciation dictionary <https://dictionary.cambridge.org/>
2. Fluentu website <https://www.fluentu.com/>
3. Repeat after us <https://brycs.org/clearinghouse/3018/>
4. Language lab <https://brycs.org/clearinghouse/3018/>
5. Oxford online videos

VIII. MATERIALS ONLINE

1. Course template
2. Lab manual



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

ELECTRICAL AND ELECTRONICS ENGINEERING LABORATORY								
I Semester: CSE (AI&ML) / IT / AERO / MECH / CIVIL								
II Semester: CSE / CSE(DS) / CSE(CS)								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AEED03	Foundation	L	T	P	C	CIA	SEE	Total
		-	-	2	1	40	60	100
Contact Classes: Nil	Tutorial Classes: Nil	Practical Classes: 45			Total Classes: 45			
Prerequisite: Nil								

I. COURSE OVERVIEW:

This course **serves** as a foundation course on electrical engineering. It **covers** a broad range of fundamental electrical circuits and devices. The **concepts** of current, voltage, power, basic circuit elements, electrical and electronic devices and their **application** in more complex electrical systems are to be imparted to the students.

II. COURSES OBJECTIVES:

The students will try to learn:

- I. The basic laws for different circuits.
- II. The elementary experimental and modeling skills for handling problems with electrical machines in the industries and domestic applications to excel in professional career.
- III. The intuitive knowledge needed to test and test and analyze the performance leading to design of electric machines by conducting various tests and calculate the performance parameters.
- IV. The semiconductor devices like diode and transistor.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO1** Solve an electric circuit by providing laws and solving theorems.
- CO2** Analyze the performance characteristics of DC shunt machine at various loading conditions
- CO3** Examine the performance of induction motors by conducting a suitable test.
- CO4** Acquire basic knowledge on the working of diodes to plot their characteristics
- CO5** Identify transistor configuration and their working to deduce its working.
- CO6** Use of the two port parameters to be measured easily, without solving for all the internal voltages and currents in the different networks.

Dos

- 1) For safety purpose the students should compulsorily wear leather shoes.
- 2) Students should come in uniform prescribed.
 - i. For boys, half sleeve shirts, tucked in trousers
 - ii. For ladies, half sleeve overcoat, hair put inside the overcoat
- 3) After giving connections, staff members should be asked to verify the circuit connections.
- 4) Before starting the circuit connections check whether the circuit breaker is in OFF condition.
- 5) Circuit should be switched ON only after getting permission from the staff member.
- 6) To be careful with moving parts in the machine.
- 7) To come prepared with procedure relevant to the experiment.
- 8) Unplug electrical equipment after use.

Don't's

- 1) Don't assume that the power is disconnected.
- 2) Don't attempt to repair electrical equipment.
- 3) Don't come with any ornaments when working with electrical machines.
- 4) Don't use an earth connection as a neutral.
- 5) Don't touch any parts unnecessarily.
- 6) Don't keep any fluids and chemicals nearing instruments and circuits.

IV. COURSE CONTENT:

EXERCISES FOR ELECTRICAL AND ELECTRICAL ENGINEERING LABORATORY

Note: Students are encouraged to bring their own laptops for laboratory practice session

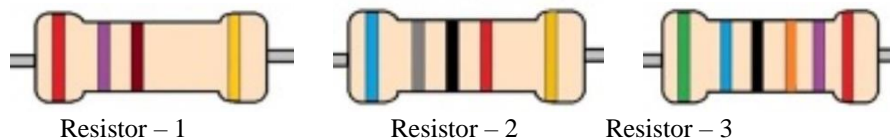
1. Getting Started Exercises

1.1 Introduction to electrical circuits

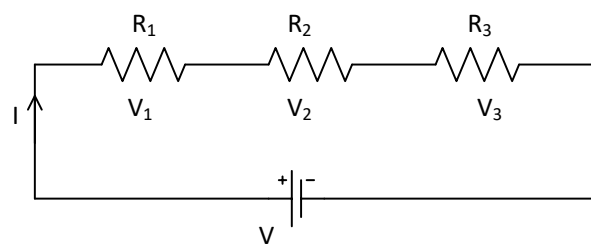
1. Understand the basic electrical equipment's used in the laboratory.
2. Become familiar with the operation and usage of basic DC electrical laboratory devices, namely DC power supplies and digital multimeter's.
3. Learn the measurement of resistance values using colour code and digital multimeter.
4. Learn the basics of circuit design using Simulink.

Try

1. Calculate the resistance value of Resistor – 1, Resistor – 2 and Resistor – 3 using colour code and verify using a digital multimeter.

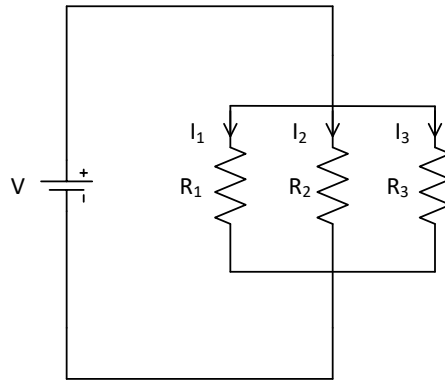


2. Design Circuit – 1 using Simulink and find the voltages V_1 , V_2 , V_3 and Current I . Where $V_s = 6\text{ V}$, $R_1 = 100\ \Omega$, $R_2 = 220\ \Omega$, $R_3 = 1\text{ k}\Omega$.



Circuit – 1

3. Design Circuit – 2 using Simulink and find the currents I_1 , I_2 and I_3 . Where $V_s = 6\text{ V}$, $R_1 = 100\ \Omega$, $R_2 = 220\ \Omega$, $R_3 = 1\text{ k}\Omega$.

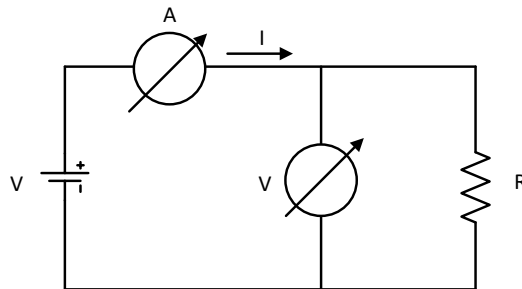


Circuit – 2

2. Exercises on Basic Electrical Circuit Law's

2.1 Ohm's law

1. Examine Ohm's law of Circuit – 3 and draw the V-I characteristic of linear resistors $R = 1\text{ k}\Omega$



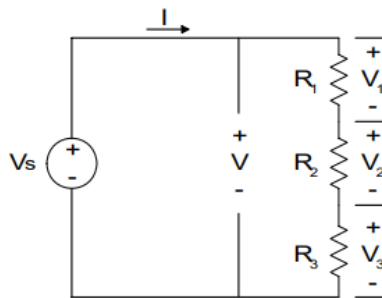
Circuit - 3

Try

1. Verify Ohm's law of Circuit – 3 using Simulink and draw the V-I characteristics of a linear resistor $R = 470\text{ k}\Omega$
2. An electric heater takes 1.48 kW from a voltage source of 220 V . Find the resistance of the heater.

2.2 Kirchhoff's voltage law

1. Examine Kirchhoff's voltage law using basic series DC Circuit - 4 with resistors. Where $V_s = 6\text{ V}$, $R_1 = 100\Omega$, $R_2 = 220\Omega$, $R_3 = 1\text{ k}\Omega$.



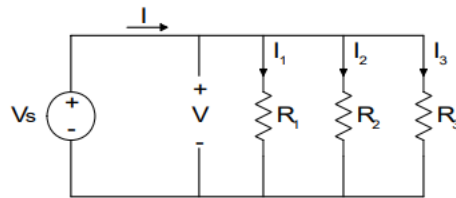
Circuit – 4

Try

1. Design and Verify Kirchhoff's voltage law for Circuit – 4 using Simulink.
2. Determine the voltage V_2 by replacing the resistor $R_2 = 150\Omega$.
3. Find the total current I flowing through the Circuit – 4.

2.3 Kirchhoff's current law

1. Examine Kirchhoff's current law using basic parallel DC Circuits - 5 with resistors. Where $V_s = 6\text{ V}$, $R_1 = 100\Omega$, $R_2 = 220\Omega$, $R_3 = 1\text{ k}\Omega$



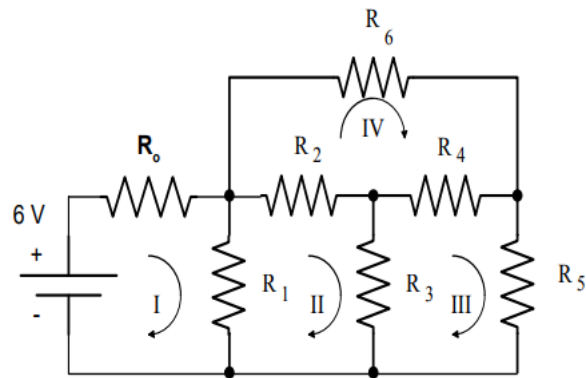
Circuit – 5

Try

1. Design and Verify Kirchhoff's current law for Circuit – 5 using Simulink.
2. Determine the voltage I_2 by replacing the resistor $R_2 = 150 \Omega$.

3. Exercises on Mesh Analysis

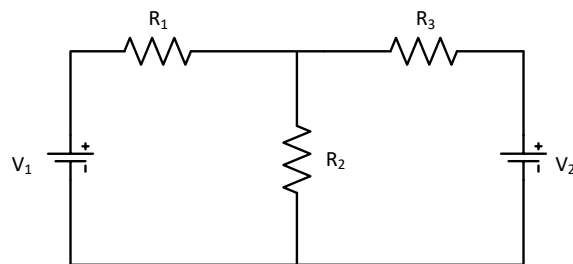
1. Determine mesh currents in the complex electrical Circuit - 6 by using principles of basic electrical circuits. Where $R_0 = 47\Omega$, $R_1 = 100 \Omega$, $R_2 = 220 \Omega$, $R_3 = 1k \Omega$, $R_4 = 150 \Omega$, $R_5 = 82 \Omega$, $R_6 = 100 \Omega$.



Circuit – 6

Try

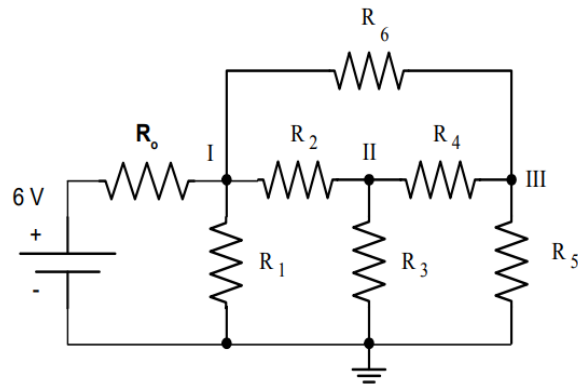
1. Use Simulink to simulate the Circuit – 6 for determining the current flowing through each resistor and compare these values to those you obtained from your experiments.
2. Find the current flowing through the resistor R_2 and R_3 by replacing the values of the resistors $R_2 = 100 \Omega$, $R_3 = 470 \Omega$.
3. Find the current flowing through the resistor R_3 for Circuit – 7 using mesh analysis when $V_1 = 10V$, $V_2 = 6V$, $R_1 = 100 \Omega$, $R_2 = 220 \Omega$ and $R_3 = 1k \Omega$.



Circuit – 7

4. Exercises on Nodal Analysis

1. Determine nodal voltages in complex electrical Circuit – 8 by using principles of basic electrical circuits. Where $R_0 = 47\Omega$, $R_1 = 100 \Omega$, $R_2 = 220 \Omega$, $R_3 = 1k \Omega$, $R_4 = 150 \Omega$, $R_5 = 82 \Omega$, $R_6 = 100 \Omega$.



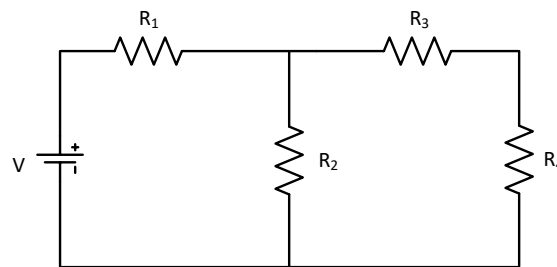
Circuit – 8

Try

1. Use Simulink to simulate the Circuit – 8 for determining the voltage across each resistor and compare these values to those you obtained from your experiments
2. Determine the voltage at node - II by replacing the resistor $R_2 = 150\ \Omega$
3. Find the voltage V_1 for Circuit – 7 using nodal analysis when $V_1 = 10V$, $V_2 = 6V$, $R_1 = 100\ \Omega$, $R_2 = 220\ \Omega$ and $R_3 = 1k\ \Omega$.

5. Exercises on Thevenin's Theorem

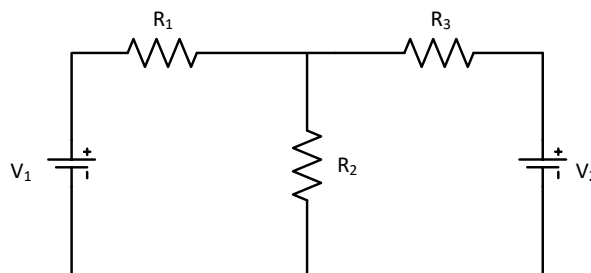
1. Determine Thevenin's equivalent voltage (V_{th}) and resistance (R_{th}) at the load terminals by applying Thevenin's theorem for Circuit – 9.
2. Determine load or unknown current through a R_4 resistor using Thevenin's equivalent circuit. Where $V = 10V$, $R_1 = 100\ \Omega$, $R_2 = 220\ \Omega$, $R_3 = 1k\ \Omega$ and $R_4 = 150\ \Omega$



Circuit – 9

Try

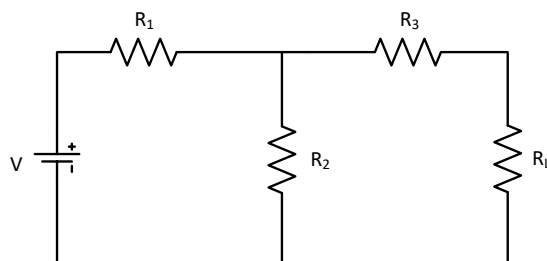
1. Use Simulink to simulate the Circuit – 13 for determining the current through R_4 resistor using Thevenin's theorem and compare this value to those you obtained from your experiment.
2. Find the current through R_4 resistor using any circuit reduction technique and verify this value to those you obtained from Thevenin's theorem.
Find the current flowing through R_2 resistor in Circuit – 10 using Thevenin's theorem for the below circuit. Where $V_1 = 10V$, $V_2 = 5V$, $R_1 = 100\ \Omega$, $R_2 = 220\ \Omega$, $R_3 = 1k\ \Omega$



Circuit – 10

6. Exercises on Norton's Theorem

1. Find Norton equivalent current (I_N) and resistance (R_N) by considering $R_L = 150\ \Omega$ resistor for the Circuit – 15 by applying Norton's theorem.
2. Find load or unknown current through R_L resistor using Norton's equivalent circuit. Where $V = 10V$, $R_1 = 100\ \Omega$, $R_2 = 220\ \Omega$ and $R_3 = 1k\ \Omega$.



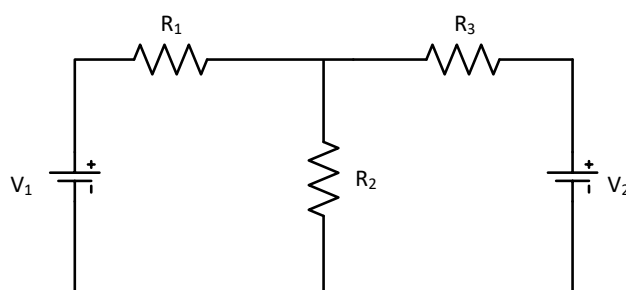
Circuit – 11

Try

1. Use Simulink to simulate the Circuit – 11 for determining the current through R_L resistor and compare this value to those you obtained from your experiment
2. Find the current through R_L using any circuit reduction technique and verify this value to those you obtained from Norton's theorem.

7. Exercises on Superposition Theorem

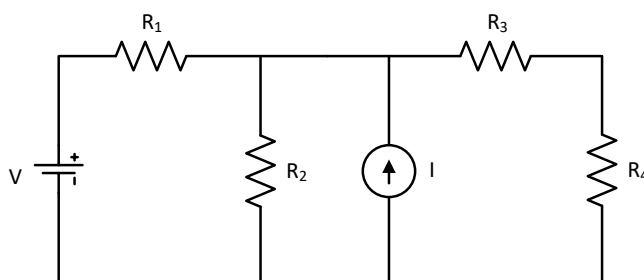
1. Investigate the current through R_2 resistor using superposition theorem to multiple DC source
Circuit - 16



Circuit – 12

Try

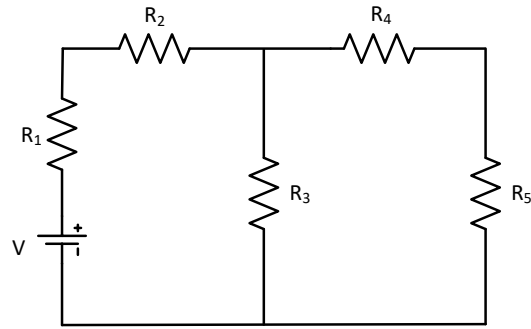
1. Use Simulink to simulate the Circuit – 12 for determining the current through a R_2 resistor and compare this value to those you obtained from your experiment
2. Find the current through R_2 in Circuit – 12 using any circuit reduction technique and verify this value to those you obtained from the superposition theorem.
3. Find the current through the R_4 resistor using the superposition theorem for the Circuit – 13.



Circuit – 13

8. Exercises on Reciprocity Theorem

1. Understand the reciprocity theorem by analyzing Circuit – 14 with interconnected components using fundamental circuit laws where $V = 10\text{ V}$, $R_1 = 100\ \Omega$, $R_2 = 220\ \Omega$, $R_3 = 1\text{ k}\Omega$, $R_4 = 150\ \Omega$ and $R_5 = 82\ \Omega$.



Circuit – 14

Try

1. Use Simulink to simulate Circuit-14 for determining the current through R_5 and R_1 by interchanging the voltage source in series with the R_5 resistor and compare this value to those you obtained from your experiment
2. Find the current through R_3 using Thevenin's theorem.

9. Swinburne's test and speed control of dc shunt motor

1. Design the suitable test under no load conditions to measure no load losses in Dc shunt machines and speed control of DC shunt motor.

Try

1. Calculate the output power and efficiency when motor takes 10A on full load and 5A on half Load.
2. Measure the no load machine losses by using indirect method of testing.
3. Perform the speed control by varying the armature circuit resistance and field circuit resistance of DC shunt motor.

10 magnetization characteristics of dc shunt generator

1. Develop the circuit for analyzing the magnetization characteristics of DC shunt generator.

Try

1. From the Open circuit characteristics calculate the critical resistance of field winding.
2. Using magnetization characteristics calculate the critical speed of DC shunt generator at $100\ \Omega$
3. Determine the performance of DC generator using the magnetization curve.
4. Calculate the critical value of shunt field resistance at 1500 rpm

11 Exercises on PN junction diode characteristics

Study the characteristics of PN junction diode as shown in Figure. 1

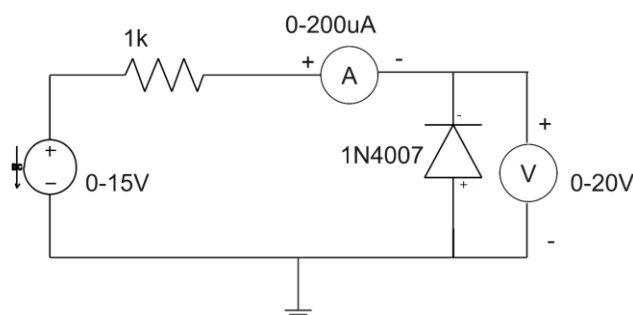


Figure. 1 diode as forward bias

Try

1. Plot the V-I Characteristics of germanium diode and find the cut in voltage.
2. Design diode acts as switch and plot the switching times of diode.

12 Zener diode characteristics and voltage regulator

Study the characteristics of Zener diode as shown in Figure. 2

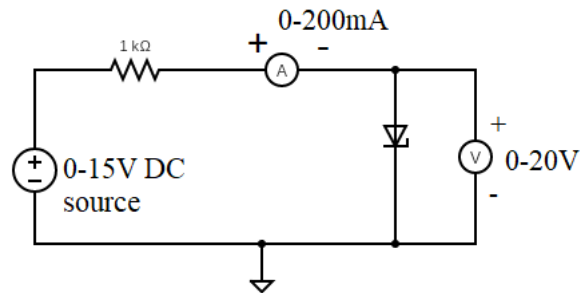


Figure. 2 Circuit diagram for Zener diode as forward bias

Try

1. Design a Zener voltage regulator circuit to drive a load of 6V, 100 mW from an unregulated
2. input supply of $V_{\min} = 8V$, $V_{\max} = 12V$ using a 6V Zener diode. .
3. Design square wave generator using Zener diode
4. Design for a Zener Transistor series voltage regulator circuit to drive a load of 6V, 1w,
5. from a supply of 10V with a $\pm 3V$ ripple voltage

13 Half wave rectifier with/without filter

1. Design a half-wave rectifier circuit and analyze its output as shown in Figure. 3
2. Analyze the rectifier output using a capacitor in shunt as a filter as shown in Figure. 3

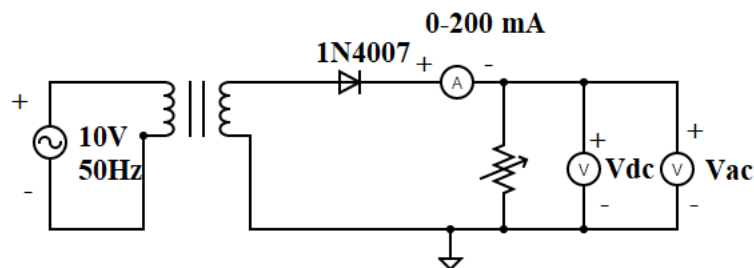


Figure. 3 Circuit Diagram for Half Wave Rectifier without Filter

Try

1. Design half wave rectifier with an applied input AC power is 100 watts, and it is to deliver an output power is 40 watts.
2. Design half wave rectifier with an AC supply of 230 V is applied through a transformer of turn ratio 10 : 1. Observe the output DC voltage, peak inverse voltage and identify dc output voltage if transformer turns ratio changed to 20:1.

14 Full wave rectifier with/without filter

1. Design a Full-wave rectifier circuit and analyse its output as shown in Figure. 4
2. Analyse the rectifier output using a capacitor in shunt as a filter as shown in Figure. 4

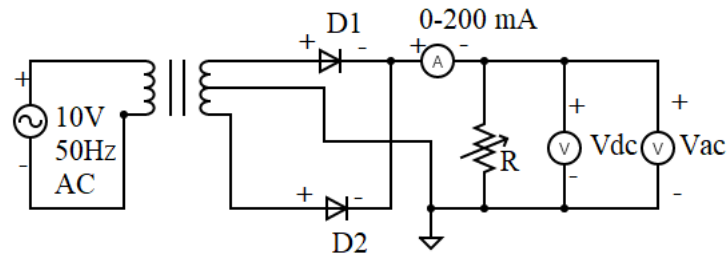


Figure 4: Circuit Diagram for Full Wave Rectifier without Filter

Try

1. Design a full wave rectifier with step down transformer and center tapped transformer. Justify the operation.
2. Design Full wave rectifier with capacitive filter using 10uF and 1uF. Observe the ripple

V. TEXT BOOKS:

1. A Chakrabarti, *CircuitTheory*, Dhanpat Rai Publications, 2004.

VI. REFERENCE BOOKS:

1. J P J Millman, C C Halkias, Satyabrata Jit, *Millman's Electronic Devices and Circuits*, Tata McGraw Hill, 2nd Edition, 1998.
2. RL Boylestad, Louis Nashelsky, *Electronic Devices and Circuits*, PEI/PHI, 9th Edition, 2006.

VII. ELECTRONICS RESOURCES:

1. <https://www.nptel.ac.in/Courses/117106108>
2. <https://www.gnindia.dronacharya.info/EEEDept/labmanuals.html>
3. <https://www.textofvideo.nptel.iitm.ac.in>
4. <https://www.textofvideo.nptel.iitm.ac.in/>

VIII. MATERIALS ONLINE

11. Course template
12. Lab manual



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

OBJECT ORIENTED PROGRAMMING WITH JAVA LABORATORY								
I Semester: AE / ME / CE / ECE / EEE / CSE / CSE (AI & ML) / CSE (DS) / CSE (CS) / IT								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
ACSD02	Foundation	L	T	P	C	CIA	SEE	Total
		0	1	2	2	40	60	100
Contact Classes: 15	Tutorial Classes: NIL	Practical Classes: 30			Total Classes: 45			
Prerequisite: There are no prerequisites to take this course.								

I. COURSE OVERVIEW:

This course provides a solid foundation in object-oriented programming concepts and hands-on experience in using them. It introduces the concepts of abstraction and reusable code design via the object-oriented paradigm. Through a series of examples and exercises students gain coding skills and develop an understanding of professional programming practices. Mastering Java facilitate the learning of other technologies.

II. COURSES OBJECTIVES:

The students will try to learn

- I. The strong foundation with the Java Virtual Machine, its concepts and features.
- II. The systematic understanding of key aspects of the Java Class Library
- III. The usage of a modern IDE with an object oriented programming language to develop programs.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO 1 Develop non-trivial programs in an modern programming language.
- CO 2 Apply the principles of selection and iteration.
- CO 3 Appreciate uses of modular programming concepts for handling complex problems.
- CO 4 Recognize and apply principle features of object-oriented design such as abstraction and encapsulation.
- CO 5 Design classes with a view of flexibility and reusability.
- CO 6 Code, test and evaluate small use cases to conform to a specification.

IV. COURE CONTENT:

EXERCISES FOR OBJECT ORIENTED PROGRAMMING WITH JAVA LABORATORY

Note: Students are encouraged to bring their own laptops for laboratory practice sessions.

1. Getting Started Exercises

1.1 HelloWorld

1. Install JDK on your machine.
2. Write a Hello-world program using JDK and a source-code editor, such as:
 - For All Platforms: Sublime Text, Atom
 - For Windows: TextPad, NotePad++
 - For macOS: jEdit, gedit
 - For Ubuntu: gedit
3. Do ALL the exercises.

1.2 Writing Good Programs

The only way to learn programming is program, program and program. Learning programming is like learning cycling, swimming or any other sports. You can't learn by watching or reading books. Start to program immediately. On the other hands, to improve your programming, you need to read many books and study how the masters program.

It is easy to write programs that work. It is much harder to write programs that not only work but also easy to maintain and understood by others – I call these good programs. In the real world, writing program is not meaningful. You have to write good programs, so that others can understand and maintain your programs.

Pay particular attention to:

1. Coding Style:
 - Read "Java Code Convention" (@ <https://www.oracle.com/technetwork/java/codeconventions-150003.pdf> or google "Java Code Convention").
 - Follow the Java Naming Conventions for variables, methods, and classes STRICTLY. Use CamelCase for names. Variable and method names begin with lowercase, while class names begin with uppercase. Use nouns for variables (e.g., radius) and class names (e.g., Circle). Use verbs for methods (e.g., getArea(), isEmpty()).
 - **Use Meaningful Names:** Do not use names like a, b, c, d, x, x1, x2, and x1688 - they are meaningless. Avoid single-alphabet names like i, j, k. They are easy to type, but usually meaningless. Use single-alphabet names only when their meaning is clear, e.g., x, y, z for coordinates and i for array index. Use meaningful names like row and col (instead of x and y, i and j, x1 and x2), numStudents (not n), maxGrade, size (not n), and upperbound (not n again). Differentiate between singular and plural nouns (e.g., use books for an array of books, and book for each item).
 - Use consistent indentation and coding style. Many IDEs (such as Eclipse/NetBeans) can re-format your source codes with a single click.
2. Program Documentation: Comment! Comment! and more Comment to explain your code to other people and to yourself three days later.

1.3 Check Pass Fail (if-else)

Write a program called **CheckPassFail** which prints "PASS" if the int variable "mark" is more than or equal to 50; or prints "FAIL" otherwise. The program shall always print "DONE" before exiting.

Hints

Use >= for greater than or equal to comparison.

```
/* Trying if-else statement.
*/
public class CheckPassFail { // Save as "CheckPassFail.java"
    public static void main(String[] args) { // Program entry point
        int mark = 49; // Set the value of "mark" here!
        System.out.println("The mark is " + mark);
    }
}
```

```
// if-else statement
if ( ..... ) {
    System.out.println( ..... );
} else {
    System.out.println( ..... );
}
System.out.println( ..... );
}
}
```

Try

mark = 0, 49, 50, 51, 100 and verify your results.

Take note of the source-code **indentation!!!** Whenever you open a block with '{', indent all the statements inside the block by 3 (or 4 spaces). When the block ends, un-indent the closing '}' to align with the opening statement.

1.4 CheckOddEven (if-else)

Write a program called **CheckOddEven** which prints "Odd Number" if the int variable "number" is odd, or "Even Number" otherwise. The program shall always print "bye!" before exiting.

Hints

n is an even number if (n % 2) is 0; otherwise, it is an odd number. Use == for comparison, e.g., (n % 2) == 0.

```
/**
 * Trying if-else statement and modulus (%) operator.
 */
public class CheckOddEven { // Save as "CheckOddEven.java"
    public static void main(String[] args) { // Program entry point
        int number = 49; // Set the value of "number" here!
        System.out.println("The number is " + number);
        if ( ..... ) {
            System.out.println( ..... ); // even number
        } else {
            System.out.println( ..... ); // odd number
        }
        System.out.println( ..... );
    }
}
```

Try

number = 0, 1, 88, 99, -1, -2 and verify your results.

Again, take note of the source-code indentation! Make it a good habit to indent your code properly, for ease of reading your program.

1.5 PrintNumberInWord (nested-if, switch-case)

Write a program called **PrintNumberInWord** which prints "ONE", "TWO",... , "NINE", "OTHER" if the int variable "number" is 1, 2,... , 9, or other, respectively. Use (a) a "nested-if" statement; (b) a "switch-case-default" statement.

Hints

```
/**
 * Trying nested-if and switch-case statements.
 */
public class PrintNumberInWord { // Save as "PrintNumberInWord.java"
    public static void main(String[] args) {
        int number = 5; // Set the value of "number" here!
    }
}
```

```

// Using nested-if
if (number == 1) { // Use == for comparison
    System.out.println( ..... );
} else if ( ..... ) {
    .....
} else if ( ..... ) {
    .....
.....
.....
} else {
    .....
}

// Using switch-case-default
switch(number) {
    case 1:
        System.out.println( ..... ); break; // Don't forget the "break" after each case!
    case 2:
        System.out.println( ..... ); break;
    .....
    .....
    default: System.out.println( ..... );
}
}
}

```

Try

number = 0, 1, 2, 3, ..., 9, 10 and verify your results.

1.6 PrintDayInWord (nested-if, switch-case)

Write a program called **PrintDayInWord** which prints “Sunday”, “Monday”, ... “Saturday” if the int variable “dayNumber” is 0, 1, ..., 6, respectively. Otherwise, it shall print “Not a valid day”. Use (a) a “nested-if” statement; (b) a “switch-case-default” statement.

Try

dayNumber = 0, 1, 2, 3, 4, 5, 6, 7 and verify your results.

2. Exercises on Number Systems (for Science/Engineering Students)

To be proficient in programming, you need to be able to operate on these number systems:

1. Decimal (used by human beings for input and output)
2. Binary (used by computer for storage and processing)
3. Hexadecimal (shorthand or compact form for binary)

2.1 Exercises (Number Systems Conversion)

1. Convert the following decimal numbers into binary and hexadecimal numbers:

- a. 108
- b. 4848
- c. 9000

Convert the following binary numbers into hexadecimal and decimal numbers:

- a. 10000000
- b. 101010101010
- c. 1000011000

Convert the following hexadecimal numbers into binary and decimal numbers:

- a. 1234
- b. 80F

c. ABCDE

Convert the following decimal numbers into binary equivalent:

- a. 123.456D
- b. 19.25D

2.2 Exercise (Integer Representation)

1. What are the ranges of 8-bit, 16-bit, 32-bit and 64-bit integer, in "unsigned" and "signed" representation?
2. Give the value of 88, 0, 1, 127, and 255 in 8-bit unsigned representation.
3. Give the value of +88, -88, -1, 0, +1, -128, and +127 in 8-bit 2's complement signed representation.
4. Give the value of +88, -88, -1, 0, +1, -127, and +127 in 8-bit sign-magnitude representation.
5. Give the value of +88, -88, -1, 0, +1, -127 and +127 in 8-bit 1's complement representation.

2.3 Exercises (Floating-point Numbers)

1. Compute the largest and smallest positive numbers that can be represented in the 32-bit normalized form.
2. Compute the largest and smallest negative numbers can be represented in the 32-bit normalized form.
3. Repeat (1) for the 32-bit denormalized form.
4. Repeat (2) for the 32-bit denormalized form.

Hints:

1. Largest positive number: S=0, E=1111 1110 (254), F=111 1111 1111 1111 1111 1111.
Smallest positive number: S=0, E=0000 0001 (1), F=000 0000 0000 0000 0000 0000.
2. Same as above, but S=1.
3. Largest positive number: S=0, E=0, F=111 1111 1111 1111 1111 1111.
Smallest positive number: S=0, E=0, F=000 0000 0000 0000 0000 0001.
4. Same as above, but S=1.

2.4 Exercises (Data Representation)

For the following 16-bit codes:

```
0000 0000 0010 1010;  
1000 0000 0010 1010;
```

Give their values, if they are representing:

1. a 16-bit unsigned integer;
2. a 16-bit signed integer;
3. two 8-bit unsigned integers;
4. two 8-bit signed integers;
5. a 16-bit Unicode characters;
6. two 8-bit ISO-8859-1 characters.

3. Exercises on Decision and Loop

3.1 SumAverageRunningInt (Decision & Loop)

Write a program called **SumAverageRunningInt** to produce the sum of 1, 2, 3, ..., to 100. Store 1 and 100 in variables lowerbound and upperbound, so that we can change their values easily. Also compute and display the average.

The output shall look like:

```
The sum of 1 to 100 is 5050  
The average is 50.5
```

Hints

```
/**
 * Compute the sum and average of running integers from a lowerbound to an upperbound using loop.
 */
public class SumAverageRunningInt { // Save as "SumAverageRunningInt.java"
    public static void main (String[] args) {
        // Define variables
        int sum = 0; // The accumulated sum, init to 0
        double average; // average in double
        final int LOWERBOUND = 1;
        final int UPPERBOUND = 100;

        // Use a for-loop to sum from lowerbound to upperbound
        for (int number = LOWERBOUND; number <= UPPERBOUND; ++number) {
            // The loop index variable number = 1, 2, 3, ..., 99, 100
            sum += number; // same as "sum = sum + number"
        }
        // Compute average in double. Beware that int / int produces int!
        .....
        // Print sum and average
        .....
    }
}
```

Try

1. Modify the program to use a "while-do" loop instead of "for" loop.

```
int sum = 0;
int number = LOWERBOUND; // declare and init loop index variable
while (number <= UPPERBOUND) { // test
    sum += number;
    ++number; // update
}
```

2. Modify the program using do-while loop

```
int sum = 0;
int number = LOWERBOUND; // declare and init loop index variable
do {
    sum += number;
    ++number; // update
} while (number <= UPPERBOUND); // test
```

3. What is the difference between "for" and "while-do" loops? What is the difference between "while-do" and "do-while" loops?
4. Modify the program to sum from 111 to 8899, and compute the average. Introduce an int variable called count to count the numbers in the specified range (to be used in computing the average).
5. Modify the program to find the "sum of the squares" of all the numbers from 1 to 100, i.e. $1*1 + 2*2 + 3*3 + \dots + 100*100$.
6. Modify the program to produce two sums: sum of odd numbers and sum of even numbers from 1 to 100. Also compute their absolute difference.

3.2 Product1ToN (or Factorial) (Decision & Loop)

Write a program called Product1ToN to compute the product of integers from 1 to 10 (i.e., $1 \times 2 \times 3 \times \dots \times 10$), as an int. Take note that It is the same as factorial of N.

Hints

Declare an int variable called product, initialize to 1, to accumulate the product.

```
// Define variables
int product = 1;    // The accumulated product, init to 1
final int LOWERBOUND = 1;
final int UPPERBOUND = 10;
```

Try

1. Compute the product from 1 to 11, 1 to 12, 1 to 13 and 1 to 14. Write down the product obtained and decide if the results are correct.

HINTS: Factorial of 13 (=6227020800) is outside the range of int [-2147483648, 2147483647]. Take note that computer programs may not produce the correct result even though the code seems correct!

2. Repeat the above, but use long to store the product. Compare the products obtained with int for N=13 and N=14.

HINTS: With long, you can store factorial of up to 20.

3.3 HarmonicSum (Decision & Loop)

Write a program called **HarmonicSum** to compute the sum of a harmonic series, as shown below, where n=50000. The program shall compute the sum from left-to-right as well as from the right-to-left. Are the two sums the same? Obtain the absolute difference between these two sums and explain the difference. Which sum is more accurate?

$$Harmonic(n) = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$$

Hints

```
/**
 * Compute the sum of harmonics series from left-to-right and right-to-left.
 */
public class HarmonicSum { // Save as "HarmonicSum.java"
    public static void main (String[] args) {
        // Define variables
        final int MAX_DENOMINATOR = 50000; // Use a more meaningful name instead of n
        double sumL2R = 0.0;    // Sum from left-to-right
        double sumR2L = 0.0;    // Sum from right-to-left
        double absDiff;        // Absolute difference between the two sums

        // for-loop for summing from left-to-right
        for (int denominator = 1; denominator <= MAX_DENOMINATOR; ++denominator) {
            // denominator = 1, 2, 3, 4, 5, ..., MAX_DENOMINATOR
            .....
            // Beware that int/int gives int, e.g., 1/2 gives 0.
        }
        System.out.println("The sum from left-to-right is: " + sumL2R);

        // for-loop for summing from right-to-left
        .....

        // Find the absolute difference and display
        if (sumL2R > sumR2L) .....
        else .....
    }
}
```

3.4 ComputePI (Decision & Loop)

Write a program called **ComputePI** to compute the value of π , using the following series expansion. Use the maximum denominator (MAX_DENOMINATOR) as the terminating condition. Try MAX_DENOMINATOR of 1000, 10000, 100000, 1000000 and compare the PI obtained. Is this series suitable for computing PI? Why?

$$\pi = 4 \times \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \frac{1}{13} - \frac{1}{15} + \dots \right)$$

Hints

Add to sum if the denominator % 4 is 1, and subtract from sum if it is 3.

```
double sum = 0.0;
int MAX_DENOMINATOR = 1000; // Try 10000, 100000, 1000000
for (int denominator = 1; denominator <= MAX_DENOMINATOR; denominator += 2) {
    // denominator = 1, 3, 5, 7, ..., MAX_DENOMINATOR
    if (denominator % 4 == 1) {
        sum += .....;
    } else if (denominator % 4 == 3) {
        sum -= .....;
    } else { // remainder of 0 or 2
        System.out.println("Impossible!!!");
    }
}
.....
```

Try

1. Instead of using maximum denominator as the terminating condition, rewrite your program to use the maximum number of terms (MAX_TERM) as the terminating condition.

```
int MAX_TERM = 10000; // number of terms used in computation
int sum = 0.0;
for (int term = 1; term <= MAX_TERM; term++) {
    // term = 1, 2, 3, 4, ..., MAX_TERM
    if (term % 2 == 1) { // odd term number: add
        sum += 1.0 / (term * 2 - 1);
    } else { // even term number: subtract
        .....
    }
}
}
```

2. JDK maintains the value of π in a built-in double constant called Math.PI (=3.141592653589793). Add a statement to compare the values obtained and the Math.PI, in percents of Math.PI, i.e., (piComputed / Math.PI) * 100.

3.5 CozaLozaWoza (Decision & Loop)

Write a program called **CozaLozaWoza** which prints the numbers 1 to 110, 11 numbers per line. The program shall print "Coza" in place of the numbers which are multiples of 3, "Loza" for multiples of 5, "Woza" for multiples of 7, "CozaLoza" for multiples of 3 and 5, and so on. The output shall look like:

```
1 2 Coza 4 Loza Coza Woza 8 Coza Loza 11
Coza 13 Woza CozaLoza 16 17 Coza 19 Loza CozaWoza 22
23 Coza Loza 26 Coza Woza 29 CozaLoza 31 32 Coza
.....
```

Hints

```
public class CozaLozaWoza { // Save as "CozaLozaWoza.java"
    public static void main(String[] args) {
        final int LOWERBOUND = 1, UPPERBOUND = 110;
        for (int number = LOWERBOUND; number <= UPPERBOUND; ++number) {
            // number = LOWERBOUND+1, LOWERBOUND+2, ..., UPPERBOUND
            // Print "Coza" if number is divisible by 3
```

```

if ( ..... ) {
    System.out.print("Coza");
}
// Print "Loza" if number is divisible by 5
if ( ..... ) {
    System.out.print(.....);
}
// Print "Woza" if number is divisible by 7
.....
// Print the number if it is not divisible by 3, 5 and 7 (i.e., it has not been processed above)
if ( ..... ) {
    .....
}
// After processing the number, print a newline if number is divisible by 11;
// else print a space
if ( ..... ) {
    System.out.println(); // print newline
} else {
    System.out.print( ..... ); // print a space
}
}
}
}

```

Notes

1. You cannot use nested-if (if ... else if ... else if ... else) for this problem. It is because the tests are not mutually exclusive. For example, 15 is divisible by both 3 and 5. Nested-if is only applicable if the tests are mutually exclusive.
2. The tests above look messy. A better solution is to use a boolean flag to keep track of whether the number has been processed, as follows:

```

final int LOWERBOUND = 1, UPPERBOUND = 110;
boolean printed;
for (int number = LOWERBOUND; number <= UPPERBOUND; ++number) {
    printed = false; // init before processing each number
    // Print "Coza" if number is divisible by 3
    if ( ..... ) {
        System.out.print( ..... );
        printed = true; // processed!
    }
    // Print "Loza" if number is divisible by 5
    if ( ..... ) {
        System.out.print( ..... );
        printed = true; // processed!
    }
    // Print "Woza" if number is divisible by 7
    .....
    // Print the number if it has not been processed
    if (!printed) {
        .....
    }
    // After processing the number, print a newline if it is divisible by 11;
    // else, print a space
    .....
}

```

3.6 Fibonacci (Decision & Loop)

Write a program called **Fibonacci** to print the first 20 Fibonacci numbers $F(n)$, where $F(n)=F(n-1)+F(n-2)$ and $F(1)=F(2)=1$. Also compute their average. The output shall look like:

The first 20 Fibonacci numbers are:

1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765

The average is 885.5

Hints

```
/**
 * Print first 20 Fibonacci numbers and their average
 */
public class Fibonacci {
    public static void main (String[] args) {
        int n = 3;        // The index n for F(n), starting from n=3, as n=1 and n=2 are pre-defined
        int fn;           // F(n) to be computed
        int fnMinus1 = 1; // F(n-1), init to F(2)
        int fnMinus2 = 1; // F(n-2), init to F(1)
        int nMax = 20;    // maximum n, inclusive
        int sum = fnMinus1 + fnMinus2; // Need sum to compute average
        double average;

        System.out.println("The first " + nMax + " Fibonacci numbers are:");
        .....

        while (n <= nMax) { // n starts from 3
            // n = 3, 4, 5, ..., nMax
            // Compute F(n), print it and add to sum
            .....
            // Increment the index n and shift the numbers for the next iteration
            ++n;
            fnMinus2 = fnMinus1;
            fnMinus1 = fn;
        }

        // Compute and display the average (=sum/nMax).
        // Beware that int/int gives int.
        .....
    }
}
```

Try

1. **Tribonacci numbers are a sequence of numbers $T(n)$ similar to Fibonacci numbers, except that a number is formed by adding the three previous numbers, i.e., $T(n)=T(n-1)+T(n-2)+T(n-3)$, $T(1)=T(2)=1$, and $T(3)=2$. Write a program called **Tribonacci** to produce the first twenty Tribonacci numbers.**

3.7 ExtractDigits (Decision & Loop)

Write a program called **ExtractDigits** to extract each digit from an int, in the reverse order. For example, if the int is 15423, the output shall be "3 2 4 5 1", with a space separating the digits.

Hints

The coding pattern for extracting individual digits from an integer n is:

1. Use $(n \% 10)$ to extract the last (least-significant) digit.
2. Use $n = n / 10$ to drop the last (least-significant) digit.
3. Repeat if $(n > 0)$, i.e., more digits to extract.

Take note that n is destroyed in the process. You may need to clone a copy.

```
int n = ...;
while (n > 0) {
    int digit = n % 10; // Extract the least-significant digit
    // Print this digit
    .....
    n = n / 10; // Drop the least-significant digit and repeat the loop
}
```

4. Exercises on Input, Decision and Loop

4.1 Add2Integer (Input)

Write a program called Add2Integers that prompts user to enter two integers. The program shall read the two integers as int; compute their sum; and print the result. For example,

```
Enter first integer: 8
Enter second integer: 9
The sum is: 17
```

Hints

```
import java.util.Scanner; // For keyboard input
/**
 * 1. Prompt user for 2 integers
 * 2. Read inputs as "int"
 * 3. Compute their sum in "int"
 * 4. Print the result
 */
public class Add2Integers { // Save as "Add2Integers.java"
    public static void main (String[] args) {
        // Declare variables
        int number1, number2, sum;

        // Put up prompting messages and read inputs as "int"
        Scanner in = new Scanner(System.in); // Scan the keyboard for input
        System.out.print("Enter first integer: "); // No newline for prompting message
        number1 = in.nextInt(); // Read next input as "int"
        .....
        in.close(); // Close Scanner

        // Compute sum
        sum = .....

        // Display result
        System.out.println("The sum is: " + sum); // Print with newline
    }
}
```

4.2 SumProductMinMax3 (Arithmetic & Min/Max)

Write a program called SumProductMinMax3 that prompts user for three integers. The program shall read the inputs as int; compute the sum, product, minimum and maximum of the three integers; and print the results. For example,

```
Enter 1st integer: 8
Enter 2nd integer: 2
Enter 3rd integer: 9
The sum is: 19
The product is: 144
The min is: 2
```

The max is: 9

Hints

```
// Declare variables
int number1, number2, number3; // The 3 input integers
int sum, product, min, max;    // To compute these

// Prompt and read inputs as "int"
Scanner in = new Scanner(System.in); // Scan the keyboard
.....
.....
in.close();

// Compute sum and product
sum = .....
product = .....

// Compute min
// The "coding pattern" for computing min is:
// 1. Set min to the first item
// 2. Compare current min with the second item and update min if second item is smaller
// 3. Repeat for the next item
min = number1;    // Assume min is the 1st item
if (number2 < min) { // Check if the 2nd item is smaller than current min
    min = number2;  // Update min if so
}
if (number3 < min) { // Continue for the next item
    min = number3;
}

// Compute max - similar to min
.....

// Print results
.....
```

Try

1. Write a program called **SumProductMinMax5** that prompts user for five integers. The program shall read the inputs as int; compute the sum, product, minimum and maximum of the five integers; and print the results. Use five int variables: number1, number2, ..., number5 to store the inputs.

4.3 CircleComputation (double & printf())

Write a program called **CircleComputation** that prompts user for the radius of a circle in floating point number. The program shall read the input as double; compute the diameter, circumference, and area of the circle in double; and print the values rounded to 2 decimal places. Use System-provided constant Math.PI for pi. The formulas are:

```
diameter = 2.0 * radius;
area = Math.PI * radius * radius;
circumference = 2.0 * Math.PI * radius;
```

Hints

```
// Declare variables
double radius, diameter, circumference, area; // inputs and results - all in double
.....

// Prompt and read inputs as "double"
System.out.print("Enter the radius: ");
```

```
radius = in.nextDouble(); // read input as double

// Compute in "double"
.....

// Print results using printf() with the following format specifiers:
// %.2f for a double with 2 decimal digits
// %n for a newline
System.out.printf("Diameter is: %.2f%n", diameter);
.....
```

Try

1. Write a program called **SphereComputation** that prompts user for the radius of a sphere in floating point number. The program shall read the input as double; compute the volume and surface area of the sphere in double; and print the values rounded to 2 decimal places. The formulas are:

```
surfaceArea = 4 * Math.PI * radius * radius;
volume = 4 / 3 * Math.PI * radius * radius * radius; // But this does not work in programming?! Why?
```

Take note that you cannot name the variable surface area with a space or surface-area with a dash. Java's naming convention is surfaceArea. Other languages recommend surface_area with an underscore.

2. Write a program called **CylinderComputation** that prompts user for the base radius and height of a cylinder in floating point number. The program shall read the inputs as double; compute the base area, surface area, and volume of the cylinder; and print the values rounded to 2 decimal places. The formulas are:

```
baseArea = Math.PI * radius * radius;
surfaceArea = 2.0 * Math.PI * radius + 2.0 * baseArea;
volume = baseArea * height;
```

4.4 Swap2Integers

Write a program called Swap2Integers that prompts user for two integers. The program shall read the inputs as int, save in two variables called number1 and number2; swap the contents of the two variables; and print the results. For examples,

```
Enter first integer: 9
Enter second integer: -9
After the swap, first integer is: -9, second integer is: 9
```

Hints

To swap the contents of two variables x and y, you need to introduce a temporary storage, say temp, and do: temp \leftarrow x; x \leftarrow y; y \leftarrow temp.

4.5 IncomeTaxCalculator (Decision)

The progressive income tax rate is mandated as follows:

Taxable Income	Rate (%)
First \$20,000	0
Next \$20,000	10
Next \$20,000	20
The remaining	30

For example, suppose that the taxable income is \$85000, the income tax payable is $\$20000 \times 0\% + \$20000 \times 10\% + \$20000 \times 20\% + \$25000 \times 30\%$.

Write a program called **IncomeTaxCalculator** that reads the taxable income (in int). The program shall calculate the income tax payable (in double); and print the result rounded to 2 decimal places. For examples,

Enter the taxable income: **\$41234**
The income tax payable is: \$2246.80

Enter the taxable income: **\$67891**
The income tax payable is: \$8367.30

Enter the taxable income: **\$85432**
The income tax payable is: \$13629.60

Enter the taxable income: **\$12345**
The income tax payable is: \$0.00

Hints

```
// Declare constants first (variables may use these constants)
// The keyword "final" marked these as constant (i.e., cannot be changed).
// Use uppercase words joined with underscore to name constants
final double TAX_RATE_ABOVE_20K = 0.1;
final double TAX_RATE_ABOVE_40K = 0.2;
final double TAX_RATE_ABOVE_60K = 0.3;

// Declare variables
int taxableIncome;
double taxPayable;
.....

// Compute tax payable in "double" using a nested-if to handle 4 cases
if (taxableIncome <= 20000) { // [0, 20000]
    taxPayable = .....;
} else if (taxableIncome <= 40000) { // [20001, 40000]
    taxPayable = .....;
} else if (taxableIncome <= 60000) { // [40001, 60000]
    taxPayable = .....;
} else { // [60001, ]
    taxPayable = .....;
}

// Alternatively, you could use the following nested-if conditions
// but the above follows the table data
//if (taxableIncome > 60000) { // [60001, ]
//    .....
//} else if (taxableIncome > 40000) { // [40001, 60000]
//    .....
//} else if (taxableIncome > 20000) { // [20001, 40000]
//    .....
//} else { // [0, 20000]
//    .....
//}

// Print results rounded to 2 decimal places
System.out.printf("The income tax payable is: %.2f%n", ...);
```

Try

Suppose that a 10% tax rebate is announced for the income tax payable, capped at \$1,000, modify your program to handle the tax rebate. For example, suppose that the tax payable is \$12,000, the rebate is \$1,000, as 10% of \$12,000 exceed the cap.

4.6 IncomeTaxCalculatorWithSentinel (Decision & Loop)

Based on the previous exercise, write a program called `IncomeTaxCalculatorWithSentinel` which shall repeat the calculation until user enter -1. For example,

Enter the taxable income (or -1 to end): **\$41000**

The income tax payable is: \$2200.00

Enter the taxable income (or -1 to end): **\$62000**

The income tax payable is: \$6600.00

Enter the taxable income (or -1 to end): **\$73123**

The income tax payable is: \$9936.90

Enter the taxable income (or -1 to end): **\$84328**

The income tax payable is: \$13298.40

Enter the taxable income: **\$-1**

bye!

The -1 is known as the sentinel value. (Wiki: In programming, a sentinel value, also referred to as a flag value, trip value, rogue value, signal value, or dummy data, is a special value which uses its presence as a condition of termination.)

Hints

The coding pattern for handling input with sentinel value is as follows:

```
// Declare constants first
final int SENTINEL = -1; // Terminating value for input
.....
// Declare variables
int taxableIncome;
double taxPayable;
.....

// Read the first input to "seed" the while loop
System.out.print("Enter the taxable income (or -1 to end): $");
taxableIncome = in.nextInt();

while (taxableIncome != SENTINEL) {
    // Compute tax payable
    .....
    // Print result
    .....

    // Read the next input
    System.out.print("Enter the taxable income (or -1 to end): $");
    taxableIncome = in.nextInt();
    // Repeat the loop body, only if the input is not the SENTINEL value.
    // Take note that you need to repeat these two statements inside/outside the loop!
}
System.out.println("bye!");
```

Take note that we repeat the input statements inside and outside the loop. Repeating statements is NOT a good programming practice. This is because it is easy to repeat (Cntl-C/Cntl-V), but hard to maintain and synchronize the repeated statements. In this case, we have no better choices!

4.7 PensionContributionCalculatorWithSentinel (Decision & Loop)

Based on the previous PensionContributionCalculator, write a program called **PensionContributionCalculatorWithSentinel** which shall repeat the calculations until user enter -1 for the salary. For examples,

Enter the monthly salary (or -1 to end): **\$5123**

Enter the age: **21**

The employee's contribution is: \$1024.60

The employer's contribution is: \$870.91

The total contribution is: \$1895.51

Enter the monthly salary (or -1 to end): **\$5123**
Enter the age: **64**
The employee's contribution is: \$384.22
The employer's contribution is: \$461.07
The total contribution is: \$845.30

Enter the monthly salary (or -1 to end): **\$-1**
bye!

Hints

```
// Read the first input to "seed" the while loop
System.out.print("Enter the monthly salary (or -1 to end): $");
salary = in.nextInt();

while (salary != SENTINEL) {
    // Read the remaining
    System.out.print("Enter the age: ");
    age = in.nextInt();

    .....

    .....

    // Read the next input and repeat
    System.out.print("Enter the monthly salary (or -1 to end): $");
    salary = in.nextInt();
}
```

4.8 SalesTaxCalculator (Decision & Loop)

A sales tax of 7% is levied on all goods and services consumed. It is also mandatory that all the price tags should include the sales tax. For example, if an item has a price tag of \$107, the actual price is \$100 and \$7 goes to the sales tax.

Write a program using a loop to continuously input the tax-inclusive price (in double); compute the actual price and the sales tax (in double); and print the results rounded to 2 decimal places. The program shall terminate in response to input of -1; and print the total price, total actual price, and total sales tax. For examples,

Enter the tax-inclusive price in dollars (or -1 to end): **107**
Actual Price is: \$100.00, Sales Tax is: \$7.00

Enter the tax-inclusive price in dollars (or -1 to end): **214**
Actual Price is: \$200.00, Sales Tax is: \$14.00

Enter the tax-inclusive price in dollars (or -1 to end): **321**
Actual Price is: \$300.00, Sales Tax is: \$21.00

Enter the tax-inclusive price in dollars (or -1 to end): **-1**
Total Price is: \$642.00
Total Actual Price is: \$600.00
Total Sales Tax is: \$42.00

Hints

```
// Declare constants
final double SALES_TAX_RATE = 0.07;
final int SENTINEL = -1;    // Terminating value for input

// Declare variables
double price, actualPrice, salesTax; // inputs and results
double totalPrice = 0.0, totalActualPrice = 0.0, totalSalesTax = 0.0; // to accumulate
.....
```

```

// Read the first input to "seed" the while loop
System.out.print("Enter the tax-inclusive price in dollars (or -1 to end): ");
price = in.nextDouble();

while (price != SENTINEL) {
    // Compute the tax
    .....
    // Accumulate into the totals
    .....
    // Print results
    .....

    // Read the next input and repeat
    System.out.print("Enter the tax-inclusive price in dollars (or -1 to end): ");
    price = in.nextDouble();
}
// print totals
.....

```

4.9 ReverseInt (Loop with Modulus/Divide)

Write a program that prompts user for a positive integer. The program shall read the input as int; and print the "reverse" of the input integer. For examples,

Enter a positive integer: **12345**
The reverse is: 54321

Hints

Use the following coding pattern which uses a while-loop with repeated modulus/divide operations to extract and drop the last digit of a positive integer.

```

// Declare variables
int inNumber; // to be input
int inDigit; // each digit
.....

// Extract and drop the "last" digit repeatably using a while-loop with modulus/divide operations
while (inNumber > 0) {
    inDigit = inNumber % 10; // extract the "last" digit
    // Print this digit (which is extracted in reverse order)
    .....
    inNumber /= 10; // drop "last" digit and repeat
}
.....

```

4.10 SumOfDigitsInt (Loop with Modulus/Divide)

Write a program that prompts user for a positive integer. The program shall read the input as int; compute and print the sum of all its digits. For examples,

Enter a positive integer: **12345**
The sum of all digits is: 15

Hints

See "ReverseInt".

4.11 InputValidation (Loop with boolean flag)

Your program often needs to validate the user's inputs, e.g., marks shall be between 0 and 100.

Write a program that prompts user for an integer between 0-10 or 90-100. The program shall read the input as int; and repeat until the user enters a valid input. For examples,

```
Enter a number between 0-10 or 90-100: -1
Invalid input, try again...
Enter a number between 0-10 or 90-100: 50
Invalid input, try again...
Enter a number between 0-10 or 90-100: 101
Invalid input, try again...
Enter a number between 0-10 or 90-100: 95
You have entered: 95
```

Hints

Use the following coding pattern which uses a do-while loop controlled by a boolean flag to do input validation. We use a do-while instead of while-do loop as we need to execute the body to prompt and process the input at least once.

```
// Declare variables
int numberIn;    // to be input
boolean isValid; // boolean flag to control the loop
.....

// Use a do-while loop controlled by a boolean flag
// to repeatedly read the input until a valid input is entered
isValid = false; // default assuming input is not valid
do {
    // Prompt and read input
    .....

    // Validate input by setting the boolean flag accordingly
    if (numberIn ..... ) {
        isValid = true; // exit the loop
    } else {
        System.out.println(.....); // Print error message and repeat
    }
} while (!isValid);
.....
```

4.12 AverageWithInputValidation (Loop with boolean flag)

Write a program that prompts user for the mark (between 0-100 in int) of 3 students; computes the average (in double); and prints the result rounded to 2 decimal places. Your program needs to perform input validation. For examples,

```
Enter the mark (0-100) for student 1: 56
Enter the mark (0-100) for student 2: 101
Invalid input, try again...
Enter the mark (0-100) for student 2: -1
Invalid input, try again...
Enter the mark (0-100) for student 2: 99
Enter the mark (0-100) for student 3: 45
The average is: 66.67
```

Hints

```
// Declare constant
final int NUM_STUDENTS = 3;

// Declare variables
int numberIn;
```

```

boolean isValid; // boolean flag to control the input validation loop
int sum = 0;
double average;
.....

for (int studentNo = 1; studentNo <= NUM_STUDENTS; ++studentNo) {
    // Prompt user for mark with input validation
    .....
    isValid = false; // reset assuming input is not valid
    do {
        .....
    } while (!isValid);

    sum += .....;
}
.....

```

5. Exercises on Nested-Loops

5.1 SquarePattern (nested-loop)

Write a program called **SquarePattern** that prompts user for the size (a non-negative integer in int); and prints the following square pattern using two nested for-loops.

Enter the size: 5

```

#####
#####
#####
#####
#####

```

Hints

The code pattern for printing 2D patterns using nested loops is:

```

// Outer loop to print each of the rows
for (int row = 1; row <= size; row++) { // row = 1, 2, 3, ..., size
    // Inner loop to print each of the columns of a particular row
    for (int col = 1; col <= size; col++) { // col = 1, 2, 3, ..., size
        System.out.print( ..... ); // Use print() without newline inside the inner loop
        .....
    }
    // Print a newline after printing all the columns
    System.out.println();
}

```

Notes

1. You should name the loop indexes row and col, NOT i and j, or x and y, or a and b, which are meaningless.
2. The row and col could start at 1 (and upto size), or start at 0 (and upto size-1). As computer counts from 0, it is probably more efficient to start from 0. However, since humans counts from 1, it is easier to read if you start from 1.

Try

Rewrite the above program using nested while-do loops.

5.2 CheckerPattern (nested-loop)

Write a program called **CheckerPattern** that prompts user for the size (a non-negative integer in int); and prints the following checkerboard pattern.

Enter the size: 7

```
#####
#####
#####
#####
#####
#####
#####
```

Hints

```
// Outer loop to print each of the rows
for (int row = 1; row <= size; row++) { // row = 1, 2, 3, ..., size
    // Inner loop to print each of the columns of a particular row
    for (int col = 1; col <= size; col++) { // col = 1, 2, 3, ..., size
        if ((row % 2) == 0) { // row 2, 4, 6, ...
            .....
        }
        System.out.print( ..... ); // Use print() without newline inside the inner loop
        .....
    }
    // Print a newline after printing all the columns
    System.out.println();
}
```

5.3 TimeTable (nested-loop)

Write a program called **TimeTable** that prompts user for the size (a positive integer in int); and prints the multiplication table as shown:

Enter the size: 10

```
* | 1 2 3 4 5 6 7 8 9 10
-----
1 | 1 2 3 4 5 6 7 8 9 10
2 | 2 4 6 8 10 12 14 16 18 20
3 | 3 6 9 12 15 18 21 24 27 30
4 | 4 8 12 16 20 24 28 32 36 40
5 | 5 10 15 20 25 30 35 40 45 50
6 | 6 12 18 24 30 36 42 48 54 60
7 | 7 14 21 28 35 42 49 56 63 70
8 | 8 16 24 32 40 48 56 64 72 80
9 | 9 18 27 36 45 54 63 72 81 90
10 | 10 20 30 40 50 60 70 80 90 100
```

Hints

1. Use printf() to format the output, e.g., each cell is %4d.
2. See "Java Basics" article.

5.4 TriangularPattern (nested-loop)

Write 4 programs called **TriangularPatternX** (X = A, B, C, D) that prompts user for the size (a non-negative integer in int); and prints each of the patterns as shown:

Enter the size: 8

```
#           #####           #####           #
##          #####          #####          ##
###         #####         #####         ###
####        #####        #####        ####
```

#####	####	####	#####
#####	###	###	#####
#####	##	##	#####
#####	#	#	#####
(a)	(b)	(c)	(d)

Hints

1. On the main diagonal, $row = col$. On the opposite diagonal, $row + col = size + 1$, where row and col begin from 1.
2. You need to print the leading blanks, in order to push the # to the right. The trailing blanks are optional, which does not affect the pattern.
3. For pattern (a), if $(row \geq col)$ print #. Trailing blanks are optional.
4. For pattern (b), if $(row + col \leq size + 1)$ print #. Trailing blanks are optional.
5. For pattern (c), if $(row \geq col)$ print #; else print blank. Need to print the leading blanks.
6. For pattern (d), if $(row + col \geq size + 1)$ print #; else print blank. Need to print the leading blanks.
7. The coding pattern is:

```
// Outer loop to print each of the rows
for (int row = 1; row <= size; row++) { // row = 1, 2, 3, ..., size
    // Inner loop to print each of the columns of a particular row
    for (int col = 1; col <= size; col++) { // col = 1, 2, 3, ..., size
        if (.....) {
            System.out.print("# ");
        } else {
            System.out.print(" "); // Need to print the "leading" blanks
        }
    }
    // Print a newline after printing all the columns
    System.out.println();
}
```

5.5 BoxPattern (nested-loop)

Write 4 programs called **BoxPatternX** ($X = A, B, C, D$) that prompts user for the size (a non-negative integer in int); and prints the pattern as shown:

Enter the size: 8

#####	#####	#####	#####	#####
#	#	#	#	#
#	#	#	#	#
#	#	#	#	#
#	#	#	#	#
#	#	#	#	#
#####	#####	#####	#####	#####
(a)	(b)	(c)	(d)	(e)

Hints

1. On the main diagonal, $row = col$. On the opposite diagonal, $row + col = size + 1$, where row and col begin from 1.
2. For pattern (a), if $(row == 1 \parallel row == size \parallel col == 1 \parallel col == size)$ print #; else print blank. Need to print the intermediate blanks.
3. For pattern (b), if $(row == 1 \parallel row == size \parallel row == col)$ print #; else print blank.

5.6 HillPattern (nested-loop)

Write 3 programs called **HillPatternX** ($X = A, B, C, D$) that prompts user for the size (a non-negative integer in int); and prints the pattern as shown:

Enter the rows: 6

```

#          #####
###       #####
#####    #####
#####    #####
#####    #####
#####    #####
#####    #####
(a)      (b)

#####    ##    ##
#####    ##    ##
#####    ##    ##
#####    ##    ##
#####    ##    ##
#####    ##    ##
#####    ##    ##
(c)      (d)

```

Hints

For pattern (a):

```

for (int row = 1; ..... ) {
    // numCol = 2*numRows - 1
    for (int col = 1; ..... ) {
        if ((row + col >= numRows + 1) && (row >= col - numRows + 1)) {
            .....;
        } else {
            .....;
        }
    }
    .....;
}

```

or, use 2 sequential inner loops to print the columns:

```

for (int row = 1; row <= rows; row++) {
    for (int col = 1; col <= rows; col++) {
        if ((row + col >= rows + 1)) {
            .....
        } else {
            .....
        }
    }
    for (int col = 2; col <= rows; col++) { // skip col = 1
        if (row >= col) {
            .....
        } else {
            .....
        }
    }
    .....
}

```

5.7 NumberPattern (nested-loop)

Write 4 programs called **NumberPatternX** (X = A, B, C, D) that prompts user for the size (a non-negative integer in int); and prints the pattern as shown:

Enter the size: 8

```

1          1 2 3 4 5 6 7 8          1    8 7 6 5 4 3 2 1
1 2        1 2 3 4 5 6 7          2 1    7 6 5 4 3 2 1

```

1 2 3	1 2 3 4 5 6	3 2 1	6 5 4 3 2 1
1 2 3 4	1 2 3 4 5	4 3 2 1	5 4 3 2 1
1 2 3 4 5	1 2 3 4	5 4 3 2 1	4 3 2 1
1 2 3 4 5 6	1 2 3	6 5 4 3 2 1	3 2 1
1 2 3 4 5 6 7	1 2	7 6 5 4 3 2 1	2 1
1 2 3 4 5 6 7 8	1	8 7 6 5 4 3 2 1	1
(a)	(b)	(c)	(d)

6. Magic(Special) Numbers

6.1. Amicable umbers

Two different numbers are said to be so Amicable numbers if each sum of divisors is equal to the other number. Amicable Numbers are: (220, 284), (1184, 1210), (2620, 2924), (5020, 5564), (6232, 6368). For example,

Enter 1st number: 228
Enter 2nd number: 220
The numbers are Amicable Numbers.

Hints

220 and 284 are Amicable Numbers.

Divisors of 220 = 1, 2, 4, 5, 10, 11, 20, 22, 44, 55, 110

$1+2+4+5+10+11+20+22+44+55+110 = 284$

Divisors of 284 = 1, 2, 4, 71, 142

$1+2+4+71+142 = 220$

6.2. Armstrong Number

Armstrong number is a positive number if it is equal to the sum of cubes of its digits is called Armstrong number and if its sum is not equal to the number then it's not a Armstrong number. For example,

Enter number=145
145 is not an Armstrong Number

Enter number = 153
153 is an Armstrong Number

Hints

Examples: 153 is Armstrong

$(1*1*1)+(5*5*5)+(3*3*3) = 153$

6.3. Capricorn Number

A number is called Capricorn or Kaprekar number whose square is divided into two parts in any conditions and parts are added, the additions of parts is equal to the number, is called Capricorn or Kaprekar number. For example,

Enter a number : 45
45 is a Capricorn/Kaprekar number
Enter a number : 297
297 is a Capricorn/Kaprekar number
Enter a number : 44
44 is not a Capricorn/Kaprekar number

Hints

Number = 45
 $(45)^2 = 2025$

All parts for 2025:
 $202 + 5 = 207$ (not 45)
 $20 + 25 = 45$
 $2 + 025 = 27$ (not 45)

From the above we can see one combination is equal to number so that 45 is Capricorn or Kaprekar number.

Try

Write a Java program to generate and show all Kaprekar numbers less than 1000.

6.4. Circular Prime

A circular prime is a prime number with the property that the number generated at each intermediate step when cyclically permuting its digits will be prime. For example, 1193 is a circular prime, since 1931, 9311 and 3119 all are also prime. For example,

Enter a number : 137
137 is a Circular Prime
Enter a number : 44
44 is not a Circular Prime

6.5. Happy Number

A happy number is a natural number in a given number base that eventually reaches 1 when iterated over the perfect digital invariant function for. Those numbers that do not end in 1 are -unhappy numbers. For example,

Enter a number : 31
31 is a Happy number

Enter a number : 32
32 is not a Happy number

6.6. Automorphic Number

An Automorphic number is a number whose square “ends” in the same digits as the number itself. For example,

Enter a number : 5
5 is a Automorphic Number

Enter a number : 25
25 is a Automorphic Number

Enter a number : 2
2 is not a Automorphic Number

Hints

$5*5 = 25$, $6*6 = 36$, $25*25 = 625$

5,6,25 are automorphic numbers

6.7. Disarium Number

A number is called Disarium number if the sum of its power of the positions from left to right is equal to the number. For example,

Enter a number : 135
135 is a Disarium Number

Enter a number : 32
32 is not a Disarium Number

Hints

$$1^1 + 3^2 + 5^3 = 1 + 9 + 125 = 135$$

6.8. Magic Number

Magic number is the if the sum of its digits recursively are calculated till a single digit If the single digit is 1 then the number is a magic number. Magic number is very similar with Happy Number. For example,

Enter a number : 226
226 is a Magic Number

Enter a number : 32
32 is not a Magic Number
Enter number = 541
153 is a Magic Number

Hints

226 is said to be a magic number

$2+2+6=10$ sum of digits is 10 then again $1+0=1$ now we get a single digit number is 1. if we single digit number will now 1 then it would not a magic number.

6.9. Neon Number

A neon number is a number where the sum of digits of square of the number is equal to the number. For example if the input number is 9, its square is $9*9 = 81$ and sum of the digits is 9. i.e. 9 is a neon number. For example,

Enter a number: 9
9 is a Neon Number

Enter a number: 8
8 is not a Neon Number

6.10. Palindromic Number

A palindromic number is a number that remains the same when its digits are reversed. For example,

Enter a number : 16461
16461 is a Palendromic Number

Enter a number : 1234
1234 is not a Plaindromic Number

6.11. Perfect Number

A perfect number is a positive integer that is equal to the sum of its positive divisors, excluding the number itself. For instance, 6 has divisors 1, 2 and 3, and $1 + 2 + 3 = 6$, so 6 is a perfect number. For example,

Enter a number : 6
6 is a Perfect Number

Enter a number : 3
3 is not a Perfect Number

6.12. Special Number

A number is said to be special number when the sum of factorial of its digits is equal to the number itself.

Example- 145 is a Special Number as $1!+4!+5!=145$. For example,

```
Enter a number : 145
145 is a Special Number
```

```
Enter a number : 23
23 is not a Special Number
```

6.13. Spy Number

A spy number is a number where the sum of its digits equals the product of its digits. For example, 1124 is a spy number, the sum of its digits is $1+1+2+4=8$ and the product of its digits is $1*1*2*4=8$. For example,

```
Enter a number : 1124
1124 is a Spy Number
```

```
Enter a number : 12
12 is not a Spy Number
```

6.14. Ugly Number

A number is said to be an Ugly number if positive numbers whose prime factors only include 2, 3, 5. For example, $6(2 \times 3)$, $8(2 \times 2 \times 2)$, $15(3 \times 5)$ are ugly numbers while $14(2 \times 7)$ is not ugly since it includes another prime factor 7. Note that 1 is typically treated as an ugly number. For example,

```
Enter a number : 6
6 is an Ugly Number
```

```
Enter a number : 14
14 is not an Ugly Number
```

7. Exercises on String and char Operations

7.1 ReverseString (String & char)

Write a program called **ReverseString**, which prompts user for a String, and prints the reverse of the String by extracting and processing each character. The output shall look like:

```
Enter a String: abcdef
The reverse of the String "abcdef" is "fedcba".
```

Hints

For a String called inStr, you can use inStr.length() to get the length of the String; and inStr.charAt(idx) to retrieve the char at the idx position, where idx begins at 0, up to inStr.length() - 1.

```
// Define variables
String inStr;    // input String
int inStrLen;    // length of the input String
.....

// Prompt and read input as "String"
System.out.print("Enter a String: ");
inStr = in.next(); // use next() to read a String
inStrLen = inStr.length();

// Use inStr.charAt(index) in a loop to extract each character
// The String's index begins at 0 from the left.
// Process the String from the right
for (int charIdx = inStrLen - 1; charIdx >= 0; --charIdx) {
    // charIdx = inStrLen-1, inStrLen-2, ... ,0
    .....
}
```

7.2 CountVowelsDigits (String & char)

Write a program called **CountVowelsDigits**, which prompts the user for a String, counts the number of vowels (a, e, i, o, u, A, E, I, O, U) and digits (0-9) contained in the string, and prints the counts and the percentages (rounded to 2 decimal places). For example,

```
Enter a String: testing12345
Number of vowels: 2 (16.67%)
Number of digits: 5 (41.67%)
```

Hints

1. To check if a char c is a digit, you can use boolean expression (c >= '0' && c <= '9'); or use built-in boolean function Character.isDigit(c).
2. You could use in.next().toLowerCase() to convert the input String to lowercase to reduce the number of cases.
3. To print a % using printf(), you need to use %%. This is because % is a prefix for format specifier in printf(), e.g., %d and %f.

7.3 PhoneKeyPad (String & char)

On you phone keypad, the alphabets are mapped to digits as follows:

ABC(2), DEF(3), GHI(4), JKL(5), MNO(6), PQRS(7), TUV(8), WXYZ(9).

Write a program called **PhoneKeyPad**, which prompts user for a String (case insensitive), and converts to a sequence of keypad digits. Use (a) a nested-if, (b) a switch-case-default.

Hints

1. You can use in.next().toLowerCase() to read a String and convert it to lowercase to reduce your cases.
2. In switch-case, you can handle multiple cases by omitting the break statement, e.g.,

```
switch (inChar) {
    case 'a': case 'b': case 'c': // No break for 'a' and 'b', fall thru 'c'
        System.out.print(2); break;
    case 'd': case 'e': case 'f':
        .....
    default:
        .....
}
```

7.4 Caesar's Code (String & char)

Caesar's Code is one of the simplest encryption techniques. Each letter in the plaintext is replaced by a letter some fixed number of position (n) down the alphabet cyclically. In this exercise, we shall pick n=3. That is, 'A' is replaced by 'D', 'B' by 'E', 'C' by 'F', ..., 'X' by 'A', ..., 'Z' by 'C'.

Write a program called **CaesarCode** to cipher the Caesar's code. The program shall prompt user for a plaintext string consisting of mix-case letters only; compute the ciphertext; and print the ciphertext in uppercase. For example,

```
Enter a plaintext string: Testing
The ciphertext string is: WHVWLQJ
```

Hints

1. Use `in.next().toUpperCase()` to read an input string and convert it into uppercase to reduce the number of cases.
2. You can use a big nested-if with 26 cases ('A'-'Z'). But it is much better to consider 'A' to 'W' as one case; 'X', 'Y' and 'Z' as 3 separate cases.
3. Take note that char 'A' is represented as Unicode number 65 and char 'D' as 68. However, 'A' + 3 gives 68. This is because `char + int` is implicitly casted to `int + int` which returns an int value. To obtain a char value, you need to perform explicit type casting using `(char)('A' + 3)`. Try printing ('A' + 3) with and without type casting.

7.5 Decipher Caesar's Code (String & char)

Write a program called **DecipherCaesarCode** to decipher the Caesar's code described in the previous exercise. The program shall prompts user for a ciphertext string consisting of mix-case letters only; compute the plaintext; and print the plaintext in uppercase. For example,

Enter a ciphertext string: **wHVwLQJ**
The plaintext string is: TESTING

7.6 Exchange Cipher (String & char)

This simple cipher exchanges 'A' and 'Z', 'B' and 'Y', 'C' and 'X', and so on.

Write a program called **ExchangeCipher** that prompts user for a plaintext string consisting of mix-case letters only. Your program shall compute the ciphertext; and print the ciphertext in uppercase. For examples,

Enter a plaintext string: **abcXYZ**
The ciphertext string is: ZYXCBA

Hints

1. Use `in.next().toUpperCase()` to read an input string and convert it into uppercase to reduce the number of cases.
2. You can use a big nested-if with 26 cases ('A'-'Z'), or use the following relationship:

'A' + 'Z' == 'B' + 'Y' == 'C' + 'X' == ... == `plainTextChar + cipherTextChar`
Hence, `cipherTextChar = 'A' + 'Z' - plainTextChar`

7.7 TestPalindromicWord and TestPalindromicPhrase (String & char)

A word that reads the same backward as forward is called a palindrome, e.g., "mom", "dad", "racecar", "madam", and "Radar" (case-insensitive).

Write a program called **TestPalindromicWord**, that prompts user for a word and prints "'xxx' is|is not a palindrome".

A phrase that reads the same backward as forward is also called a palindrome, e.g., "Madam, I'm Adam", "A man, a plan, a canal - Panama!" (ignoring punctuation and capitalization).

Modify your program (called **TestPalindromicPhrase**) to check for palindromic phrase. Use `in.nextLine()` to read a line of input.

Hints

1. Maintain two indexes, **forwardIndex (fIdx)** and **backwardIndex (bIdx)**, to scan the phrase forward and backward.

```
int fIdx = 0, bIdx = strLen - 1;
while (fIdx < bIdx) {
    .....
    ++fIdx;
    --bIdx;
}
```

```

    }
    // or
    for (int fIdx = 0, bIdx = strLen - 1; fIdx < bIdx; ++fIdx, --bIdx) {
        .....
    }

```

2. You can check if a char `c` is a letter either using built-in boolean function `Character.isLetter(c)`; or boolean expression `(c >= 'a' && c <= 'z')`. Skip the index if it does not contain a letter.

7.8 CheckBinStr (String & char)

The binary number system uses 2 symbols, 0 and 1. Write a program called **CheckBinStr** to verify a binary string. The program shall prompt user for a binary string; and decide if the input string is a valid binary string. For example,

```

Enter a binary string: 10101100
"10101100" is a binary string

```

```

Enter a binary string: 10120000
"10120000" is NOT a binary string

```

Hints

Use the following coding pattern which involves a boolean flag to check the input string.

```

// Declare variables
String inStr;    // The input string
int inStrLen;    // The length of the input string
char inChar;     // Each char of the input string
boolean isValid; // "is" or "is not" a valid binary string?
.....

isValid = true; // Assume that the input is valid, unless our check fails
for (.....) {
    inChar = .....;
    if (!(inChar == '0' || inChar == '1')) {
        isValid = false;
        break; // break the loop upon first error, no need to continue for more errors
               // If this is not encountered, isValid remains true after the loop.
    }
}
if (isValid) {
    System.out.println(.....);
} else {
    System.out.println(.....);
}
// or using one liner
//System.out.println(isValid ? ... : ...);

```

7.9 CheckHexStr (String & char)

The hexadecimal (hex) number system uses 16 symbols, 0-9 and A-F (or a-f). Write a program to verify a hex string. The program shall prompt user for a hex string; and decide if the input string is a valid hex string. For examples,

```

Enter a hex string: 123aBc
"123aBc" is a hex string

```

```

Enter a hex string: 123aBcx
"123aBcx" is NOT a hex string

```

Hints

```

if (!(inChar >= '0' && inChar <= '9')

```

```

|| (inChar >= 'A' && inChar <= 'F')
|| (inChar >= 'a' && inChar <= 'f')) { // Use positive logic and then reverse
.....
}

```

7.10 Bin2Dec (String & char)

Write a program called **Bin2Dec** to convert an input binary string into its equivalent decimal number. Your output shall look like:

```

Enter a Binary string: 1011
The equivalent decimal number for binary "1011" is: 11

Enter a Binary string: 1234
error: invalid binary string "1234"

```

Hints

See "Code Example".

7.11 Hex2Dec (String & char)

Write a program called **Hex2Dec** to convert an input hexadecimal string into its equivalent decimal number. Your output shall look like:

```

Enter a Hexadecimal string: 1a
The equivalent decimal number for hexadecimal "1a" is: 26

Enter a Hexadecimal string: 1y3
error: invalid hexadecimal string "1y3"

```

Hints

See "Code Example".

7.12 Oct2Dec (String & char)

Write a program called **Oct2Dec** to convert an input Octal string into its equivalent decimal number. For example,

```

Enter an Octal string: 147
The equivalent decimal number "147" is: 103

```

8. Exercises on Arrays

8.1 PrintArray (Array)

Write a program called **PrintArray** which prompts user for the number of items in an array (a non-negative integer), and saves it in an int variable called **NUM_ITEMS**. It then prompts user for the values of all the items and saves them in an int array called **items**. The program shall then print the contents of the array in the form of [x1, x2, ..., xn]. For example,

```

Enter the number of items: 5
Enter the value of all items (separated by space): 3 2 5 6 9
The values are: [3, 2, 5, 6, 9]

```

Hints

```

// Declare variables
tinal int NUM_ITEMS;
int[] items; // Declare array name, to be allocated after NUM_ITEMS is known
.....

```

```

// Prompt for for the number of items and read the input as "int"
.....
NUM_ITEMS = .....

// Allocate the array
items = new int[NUM_ITEMS];

// Prompt and read the items into the "int" array, if array length > 0
if (items.length > 0) {
    .....
    for (int i = 0; i < items.length; ++i) { // Read all items
        .....
    }
}

// Print array contents, need to handle first item and subsequent items differently
.....
for (int i = 0; i < items.length; ++i) {
    if (i == 0) {
        // Print the first item without a leading commas
        .....
    } else {
        // Print the subsequent items with a leading commas
        .....
    }
}
// or, using a one liner
//System.out.print((i == 0) ? ..... : .....);
}

```

8.2 PrintArrayInStars (Array)

Write a program called **printArrayInStars** which prompts user for the number of items in an array (a non-negative integer), and saves it in an int variable called NUM_ITEMS. It then prompts user for the values of all the items (non-negative integers) and saves them in an int array called items. The program shall then print the contents of the array in a graphical form, with the array index and values represented by number of stars. For examples,

```

Enter the number of items: 5
Enter the value of all items (separated by space): 7 4 3 0 7
0: *****(7)
1: ****(4)
2: ***(3)
3: (0)
4: *****(7)

```

Hints

```

// Declare variables
final int NUM_ITEMS;
int[] items; // Declare array name, to be allocated after NUM_ITEMS is known
.....
.....
// Print array in "index: number of stars" using a nested-loop
// Take note that rows are the array indexes and columns are the value in that index
for (int idx = 0; idx < items.length; ++idx) { // row
    System.out.print(idx + ": ");
    // Print value as the number of stars
    for (int starNo = 1; starNo <= items[idx]; ++starNo) { // column
        System.out.print("*");
    }
    .....
}

```



```
}  
.....
```

8.3 GradesStatistics (Array)

Write a program which prompts user for the number of students in a class (a non-negative integer), and saves it in an int variable called numStudents. It then prompts user for the grade of each of the students (integer between 0 to 100) and saves them in an int array called grades. The program shall then compute and print the average (in double rounded to 2 decimal places) and minimum/maximum (in int).

```
Enter the number of students: 5  
Enter the grade for student 1: 98  
Enter the grade for student 2: 78  
Enter the grade for student 3: 78  
Enter the grade for student 4: 87  
Enter the grade for student 5: 76  
The average is: 83.40  
The minimum is: 76  
The maximum is: 98
```

8.4 Hex2Bin (Array for Table Lookup)

Write a program called **Hex2Bin** that prompts user for a hexadecimal string and print its equivalent binary string. The output shall look like:

```
Enter a Hexadecimal string: 1abc  
The equivalent binary for hexadecimal "1abc" is: 0001 1010 1011 1100
```

Hints

1. Use an array of 16 Strings containing binary strings corresponding to hexadecimal number 0-9A-F (or a-f), as follows

```
final String[] HEX_BITS = {"0000", "0001", "0010", "0011",  
    "0100", "0101", "0110", "0111",  
    "1000", "1001", "1010", "1011",  
    "1100", "1101", "1110", "1111"};
```

8.5 Dec2Hex (Array for Table Lookup)

Write a program called **Dec2Hex** that prompts user for a positive decimal number, read as int, and print its equivalent hexadecimal string. The output shall look like:

```
Enter a decimal number: 1234  
The equivalent hexadecimal number is 4D2
```

9. Exercises on Methods

9.1 exponent() (method)

Write a method called exponent(int base, int exp) that returns an int value of base raises to the power of exp. The signature of the method is:

```
public static int exponent(int base, int exp);
```

Assume that exp is a non-negative integer and base is an integer. Do not use any Math library functions. Also write the main() method that prompts user for the base and exp; and prints the result. For example,

```
Enter the base: 3  
Enter the exponent: 4  
3 raises to the power of 4 is: 81
```

Hints

```
.....
public class Exponent {
    public static void main(String[] args) {
        // Declare variables
        int exp; // exponent (non-negative integer)
        int base; // base (integer)

        .....
        // Prompt and read exponent and base
        .....
        // Print result
        System.out.println(base + " raises to the power of " + exp + " is: " + exponent(base, exp));
    }

    // Returns "base" raised to the power "exp"
    public static int exponent(int base, int exp) {
        int product = 1; // resulting product

        // Multiply product and base for exp number of times
        for (.....) {
            product *= base;
        }

        return product;
    }
}
```

9.2 isOdd() (method)

Write a boolean method called `isOdd()` in a class called **OddEvenTest**, which takes an `int` as input and returns true if it is odd. The signature of the method is as follows:

```
public static boolean isOdd(int number);
```

Also write the `main()` method that prompts user for a number, and prints "ODD" or "EVEN". You should test for negative input. For examples,

```
Enter a number: 9
9 is an odd number

Enter a number: 8
8 is an even number

Enter a number: -5
-5 is an odd number
```

Hints

See Notes.

9.3 hasEight() (method)

Write a boolean method called `hasEight()`, which takes an `int` as input and returns true if the number contains the digit 8 (e.g., 18, 168, 1288). The signature of the method is as follows:

```
public static boolean hasEight(int number);
```

Write a program called **MagicSum**, which prompts user for integers (or -1 to end), and produce the sum of numbers containing the digit 8. Your program should use the above methods. A sample output of the program is as follows:

```
Enter a positive integer (or -1 to end): 1
Enter a positive integer (or -1 to end): 2
```

```
Enter a positive integer (or -1 to end): 3
Enter a positive integer (or -1 to end): 8
Enter a positive integer (or -1 to end): 88
Enter a positive integer (or -1 to end): -1
The magic sum is: 96
```

Hints

The coding pattern to repeat until input is -1 (called sentinel value) is:

```
final int SENTINEL = -1; // Terminating input
int number;

// Read first input to "seed" the while loop
System.out.print("Enter a positive integer (or -1 to end): ");
number = in.nextInt();

while (number != SENTINEL) { // Repeat until input is -1
    .....
    .....

    // Read next input. Repeat if the input is not the SENTINEL
    // Take note that you need to repeat these codes!
    System.out.print("Enter a positive integer (or -1 to end): ");
    number = in.nextInt();
}
```

You can either repeatably use modulus/divide ($n\%10$ and $n=n/10$) to extract and drop each digit in int; or convert the int to String and use the String's `charAt()` to inspect each char.

9.4 `print()` (Array & Method)

Write a method called **`print()`**, which takes an int array and print its contents in the form of [a1, a2, ..., an]. Take note that there is no comma after the last element. The method's signature is as follows:

```
public static void print(int[] array);
```

Also write a test driver to test this method (you should test on empty array, one-element array, and n-element array).

How to handle double[] or float[]? You need to write a overloaded version for double[] and a overloaded version for float[], with the following signatures:

```
public static void print(double[] array)
public static void print(float[] array)
```

The above is known as method overloading, where the same method name can have many versions, differentiated by its parameter list.

Hints

For the first element, print its value; for subsequent elements, print commas followed by the value.

9.5 `arrayToString()` (Array & Method)

Write a method called **`arrayToString()`**, which takes an int array and return a String in the form of [a1, a2, ..., an]. Take note that this method returns a String, the previous exercise returns void but prints the output. The method's signature is as follows:

```
public static String arrayToString(int[] array);
```

Also write a test driver to test this method (you should test on empty array, one-element array, and n-element array).

Notes: This is similar to the built-in function `Arrays.toString()`. You could study its source code.

9.6 contains() (Array & Method)

Write a boolean method called **contains()**, which takes an array of int and an int; and returns true if the array contains the given int. The method's signature is as follows:

```
public static boolean contains(int[] array, int key);
```

Also write a test driver to test this method.

9.7 search() (Array & Method)

Write a method called **search()**, which takes an array of int and an int; and returns the array index if the array contains the given int; or -1 otherwise. The method's signature is as follows:

```
public static int search(int[] array, int key);
```

Also write a test driver to test this method.

9.8 equals() (Array & Method)

Write a boolean method called **equals()**, which takes two arrays of int and returns true if the two arrays are exactly the same (i.e., same length and same contents). The method's signature is as follows:

```
public static boolean equals(int[] array1, int[] array2)
```

Also write a test driver to test this method.

9.9 copyOf() (Array & Method)

Write a boolean method called **copyOf()**, which takes an int Array and returns a copy of the given array. The method's signature is as follows:

```
public static int[] copyOf(int[] array)
```

Also write a test driver to test this method.

Write another version for **copyOf()** which takes a second parameter to specify the length of the new array. You should truncate or pad with zero so that the new array has the required length.

```
public static int[] copyOf(int[] array, int newLength)
```

NOTES: This is similar to the built-in function `Arrays.copyOf()`.

9.10 swap() (Array & Method)

Write a method called **swap()**, which takes two arrays of int and swap their contents if they have the same length. It shall return true if the contents are successfully swapped. The method's signature is as follows:

```
public static boolean swap(int[] array1, int[] array2)
```

Also write a test driver to test this method.

Hints

You need to use a temporary location to swap two storage locations.

```
// Swap item1 and item2
int item1, item2, temp;
temp = item1;
item1 = item2;
item2 = item1;
// You CANNOT simply do: item1 = item2; item2 = item2;
```

9.11 reverse() (Array & Method)

Write a method called **reverse()**, which takes an array of int and reverse its contents. For example, the reverse of [1,2,3,4] is [4,3,2,1]. The method's signature is as follows:

```
public static void reverse(int[] array)
```

Take note that the array passed into the method can be modified by the method (this is called "pass by reference"). On the other hand, primitives passed into a method cannot be modified. This is because a clone is created and passed into the method instead of the original copy (this is called "pass by value").

Also write a test driver to test this method.

Hints

You might use two indexes in the loop, one moving forward and one moving backward to point to the two elements to be swapped.

```
for (int fIdx = 0, bIdx = array.length - 1; fIdx < bIdx; ++fIdx, --bIdx) {  
    // Swap array[fIdx] and array[bIdx]  
    // Only need to transverse half of the array elements  
}
```

You need to use a temporary location to swap two storage locations.

```
// Swap item1 and item2  
int item1, item2, temp;  
temp = item1;  
item1 = item2;  
item2 = item1;  
// You CANNOT simply do: item1 = item2; item2 = item2;
```

9.12 GradesStatistics (Array & Method)

Write a program called **GradesStatistics**, which reads in n grades (of int between 0 and 100, inclusive) and displays the average, minimum, maximum, median and standard deviation. Display the floating-point values upto 2 decimal places. Your output shall look like:

```
Enter the number of students: 4  
Enter the grade for student 1: 50  
Enter the grade for student 2: 51  
Enter the grade for student 3: 56  
Enter the grade for student 4: 53  
The grades are: [50, 51, 56, 53]  
The average is: 52.50  
The median is: 52.00  
The minimum is: 50  
The maximum is: 56  
The standard deviation is: 2.29
```

The formula for calculating standard deviation is:

$$\sigma = \sqrt{\frac{1}{n} \sum_{i=0}^{n-1} x_i^2 - \mu^2}, \text{ where } \mu \text{ is the mean}$$

Hints:

```
public class GradesStatistics {  
    public static int[] grades; // Declare an int[], to be allocated later.  
                                // This array is accessible by all the methods.  
  
    public static void main(String[] args) {  
        readGrades(); // Read and save the inputs in global int[] grades  
        System.out.println("The grades are: ");  
        print(grades);  
        System.out.println("The average is " + average(grades));  
        System.out.println("The median is " + median(grades));  
        System.out.println("The minimum is " + min(grades));  
    }  
}
```

```

        System.out.println("The maximum is " + max(grades));
        System.out.println("The standard deviation is " + stdDev(grades));
    }

    // Prompt user for the number of students and allocate the global "grades" array.
    // Then, prompt user for grade, check for valid grade, and store in "grades".
    public static void readGrades() { ..... }

    // Print the given int array in the form of [x1, x2, x3,..., xn].
    public static void print(int[] array) { ..... }

    // Return the average value of the given int[]
    public static double average(int[] array) { ..... }

    // Return the median value of the given int[]
    // Median is the center element for odd-number array,
    // or average of the two center elements for even-number array.
    // Use Arrays.sort(anArray) to sort anArray in place.
    public static double median(int[] array) { ..... }

    // Return the maximum value of the given int[]
    public static int max(int[] array) {
        int max = array[0]; // Assume that max is the first element
        // From second element, if the element is more than max, set the max to this element.
        .....
    }

    // Return the minimum value of the given int[]
    public static int min(int[] array) { ..... }

    // Return the standard deviation of the given int[]
    public static double stdDev(int[] array) { ..... }
}

```

Take note that besides readGrade() that relies on global variable grades, all the methods are self-contained general utilities that operate on any given array.

9.13 GradesHistogram (Array & Method)

Write a program called **GradesHistogram**, which reads in n grades (as in the previous exercise), and displays the horizontal and vertical histograms. For example:

```

0 - 9: ***
10 - 19: ***
20 - 29:
30 - 39:
40 - 49: *
50 - 59: *****
60 - 69:
70 - 79:
80 - 89: *
90 - 100: **

          *
          *
*   *   *
*   *   *   *
*   *   *   *   *

0-9 10-19 20-29 30-39 40-49 50-59 60-69 70-79 80-89 90-100

```

10.1 Arithmetic (Command-Line Arguments)

Write a program called **Arithmetic** that takes three command-line arguments: two integers followed by an arithmetic operator (+, -, * or /). The program shall perform the corresponding operation on the two integers and print the result. For example:

```
java Arithmetic 3 2 +
3+2=5
```

```
java Arithmetic 3 2 -
3-2=1
```

```
java Arithmetic 3 2 /
3/2=1
```

Hints

The method `main(String[] args)` takes an argument: "an array of String", which is often (but not necessary) named `args`. This parameter captures the command-line arguments supplied by the user when the program is invoked. For example, if a user invokes:

```
java Arithmetic 12345 4567 +
```

The three command-line arguments "12345", "4567" and "+" will be captured in a String array {"12345", "4567", "+"} and passed into the `main()` method as the argument `args`. That is,

```
args is: {"12345", "4567", "+"} // args is a String array
args.length is: 3                // length of the array
args[0] is: "12345"              // 1st element of the String array
args[1] is: "4567"              // 2nd element of the String array
args[2] is: "+"                 // 3rd element of the String array
args[0].length() is: 5           // length of 1st String element
args[1].length() is: 4           // length of the 2nd String element
args[2].length() is: 1           // length of the 3rd String element
```

```
public class Arithmetic {
    public static void main (String[] args) {
        int operand1, operand2;
        char theOperator;

        // Check if there are 3 command-line arguments in the
        // String[] args by using length variable of array.
        if (args.length != 3) {
            System.err.println("Usage: java Arithmetic int1 int2 op");
            return;
        }

        // Convert the 3 Strings args[0], args[1], args[2] to int and char.
        // Use the Integer.parseInt(aStr) to convert a String to an int.
        operand1 = Integer.parseInt(args[0]);
        operand2 = .....

        // Get the operator, assumed to be the first character of
        // the 3rd string. Use method charAt() of String.
        theOperator = args[2].charAt(0);
        System.out.print(args[0] + args[2] + args[1] + "=");

        switch(theOperator) {
            case ('-'): System.out.println(operand1 - operand2); break;
            case ('+'): .....
            case ('*'): .....
            case ('/'): .....
        }
    }
}
```

```

    default:
        System.err.println("Error: invalid operator!");
    }
}
}

```

Notes:

- To provide command-line arguments, use the "cmd" or "terminal" to run your program in the form "java ClassName arg1 arg2".
- To provide command-line arguments in Eclipse, right click the source code \Rightarrow "Run As" \Rightarrow "Run Configurations..." \Rightarrow Select "Main" and choose the proper main class \Rightarrow Select "Arguments" \Rightarrow Enter the command-line arguments, e.g., "3 2 +" in "Program Arguments".
- To provide command-line arguments in NetBeans, right click the "Project" name \Rightarrow "Set Configuration" \Rightarrow "Customize..." \Rightarrow Select categories "Run" \Rightarrow Enter the command-line arguments, e.g., "3 2 +" in the "Arguments" box (but make sure you select the proper Main class).

Question: Try "java Arithmetic 2 4 *" (in CMD shell and Eclipse/NetBeans) and explain the result obtained. How to resolve this problem?

In Windows' CMD shell, * is known as a wildcard character, that expands to give the list of file in the directory (called Shell Expansion). For example, "dir *.java" lists all the file with extension of ".java". You could double-quote the * to prevent shell expansion. Eclipse has a bug in handling this, even * is double-quoted. NetBeans??

SumDigits (Command-line Arguments)

Write a program called **SumDigits** to sum up the individual digits of a positive integer, given in the command line. The output shall look like:

```

java SumDigits 12345
The sum of digits = 1 + 2 + 3 + 4 + 5 = 15

```

Exercises on Recursion

In programming, a recursive function (or method) calls itself. The classical example is factorial(n), which can be defined recursively as $f(n) = n * f(n-1)$. Nonetheless, it is important to take note that a recursive function should have a terminating condition (or base case), in the case of factorial, $f(0) = 1$. Hence, the full definition is:

```

factorial(n) = 1, for n = 0
factorial(n) = n * factorial(n-1), for all n > 1

```

For example, suppose $n = 5$:

```

// Recursive call
factorial(5) = 5 * factorial(4)
factorial(4) = 4 * factorial(3)
factorial(3) = 3 * factorial(2)
factorial(2) = 2 * factorial(1)
factorial(1) = 1 * factorial(0)
factorial(0) = 1 // Base case
// Unwinding
factorial(1) = 1 * 1 = 1
factorial(2) = 2 * 1 = 2
factorial(3) = 3 * 2 = 6
factorial(4) = 4 * 6 = 24
factorial(5) = 5 * 24 = 120 (DONE)

```

10.2 Factorial Recursive

Write a recursive method called factorial() to compute the factorial of the given integer.

```

public static int factorial(int n)

```

The recursive algorithm is:


```
factorial(n) = 1, if n = 0
factorial(n) = n * factorial(n-1), if n > 0
```

Compare your code with the iterative version of the factorial():

```
factorial(n) = 1*2*3*...*n
```

Hints

Writing recursive function is straight forward. You simply translate the recursive definition into code with return.

```
// Return the factorial of the given integer, recursively
public static int factorial(int n) {
    if (n == 0) {
        return 1; // base case
    } else {
        return n * factorial(n-1); // call itself
    }
    // or one liner
    // return (n == 0) ? 1 : n*factorial(n-1);
}
```

Notes

1. Recursive version is often much shorter.
2. The recursive version uses much more computational and storage resources, and it need to save its current states before each successive recursive call, so as to unwind later.

10.3 Fibonacci (Recursive)

Write a recursive method to compute the Fibonacci number of n, defined as follows:

```
F(0) = 0
F(1) = 1
F(n) = F(n-1) + F(n-2) for n >= 2
```

Compare the recursive version with the iterative version written earlier.

Hints

```
// Translate the recursive definition into code with return statements
public static int fibonacci(int n) {
    if (n == 0) {
        return 0;
    } else if (n == 1) {
        return 1;
    } else {
        return fibonacci(n-1) + fibonacci(n-2);
    }
}
```

10.4 Length of a Running Number Sequence (Recursive)

A special number sequence is defined as follows:

```
S(1) = 1
S(2) = 12
S(3) = 123
S(4) = 1234
.....
S(9) = 123456789 // length is 9
S(10) = 12345678910 // length is 11
```

```
S(11) = 1234567891011 // length is 13
S(12) = 123456789101112 // length is 15
.....
```

Write a recursive method to compute the length of S(n), defined as follows:

```
len(1) = 1
len(n) = len(n-1) + numOfDigits(n)
```

Also write an iterative version.

10.5 GCD (Recursive)

Write a recursive method called gcd() to compute the greatest common divisor of two given integers.

```
public static void int gcd(int a, int b)

gcd(a,b) = a, if b = 0
gcd(a,b) = gcd(b, remainder(a,b)), if b > 0
```

11. More (Difficult) Exercises

11.1 JDK Source Code

Extract the source code of the class Math from the JDK source code (JDK Installed Directory \Rightarrow "lib" \Rightarrow "src.zip" \Rightarrow "java.base" \Rightarrow "java" \Rightarrow "lang" \Rightarrow "Math.java"). Study how constants such as E and PI are defined. Also study how methods such as abs(), max(), min(), toDegree(), etc, are written.

Also study the "Integer.java", "String.java".

11.2 Matrices (2D Arrays)

Similar to Math class, write a Matrix library that supports matrix operations (such as addition, subtraction, multiplication) via 2D arrays. The operations shall support both double and int. Also write a test class to exercise all the operations programmed.

Hints

```
public class Matrix {
    // Method signatures
    public static void print(int[][] m);
    public static void print(double[][] m);
    public static boolean haveSameDimension(int[][] m1, int[][] m2); // Used in add(), subtract()
    public static boolean haveSameDimension(double[][] m1, double[][] m2);
    public static int[][] add(int[][] m1, int[][] m2);
    public static double[][] add(double[][] m1, double[][] m2);
    public static int[][] subtract(int[][] m1, int[][] m2);
    public static double[][] subtract(double[][] m1, double[][] m2);
    public static int[][] multiply(int[][] m1, int[][] m2);
    public static double[][] multiply(double[][] m1, double[][] m2);
    .....
}
```

11.3 PrintAnimalPattern (Special Characters and Escape Sequences)

Write a program called **PrintAnimalPattern**, which uses println() to produce this pattern:

```
' _ '
(©©)

/=====√
/ || %% ||
* ||---||
¥¥ ¥¥
```

Hints

Use escape sequence \uhhhh where hhhh are four hex digits to display Unicode characters such as ¥ and ©. ¥ is 165 (00A5H) and © is 169 (00A9H) in both ISO-8859-1 (Latin-1) and Unicode character sets.

Double-quote (") and black-slash (\) require escape sequence inside a String. Single quote (') does not require escape sign.

Try

Print the same pattern using printf(). (Hints: Need to use %% to print a % in printf() because % is the suffix for format specifier.)

11.4 Print Patterns (nested-loop)

Write a method to print each of the followings patterns using nested loops in a class called **PrintPatterns**. The program shall prompt user for the sizde of the pattern. The signatures of the methods are:

```
public static void printPatternX(int size); // X: A, B, C,...; size is a positive integer.
```

#####	#	#
#####	###	###
#####	#####	#####
#####	#####	#####
#####	#####	#####
###	#####	#####
#	#####	#####
(a)	(b)	#####
		#####
		#####
		###
		#
		(c)

1	1 2 3 4 5 6 7 8	1	8 7 6 5 4 3 2 1
1 2	1 2 3 4 5 6 7	2 1	7 6 5 4 3 2 1
1 2 3	1 2 3 4 5 6	3 2 1	6 5 4 3 2 1
1 2 3 4	1 2 3 4 5	4 3 2 1	5 4 3 2 1
1 2 3 4 5	1 2 3 4	5 4 3 2 1	4 3 2 1
1 2 3 4 5 6	1 2 3	6 5 4 3 2 1	3 2 1
1 2 3 4 5 6 7	1 2	7 6 5 4 3 2 1	2 1
1 2 3 4 5 6 7 8	1	8 7 6 5 4 3 2 1	1
(d)	(e)	(f)	(g)

1	1 2 3 4 5 6 7 8 7 6 5 4 3 2 1
1 2 1	1 2 3 4 5 6 7 6 5 4 3 2 1
1 2 3 2 1	1 2 3 4 5 6 5 4 3 2 1
1 2 3 4 3 2 1	1 2 3 4 5 4 3 2 1
1 2 3 4 5 4 3 2 1	1 2 3 4 3 2 1
1 2 3 4 5 6 5 4 3 2 1	1 2 3 2 1
1 2 3 4 5 6 7 6 5 4 3 2 1	1 2 1

```

1 2 3 4 5 6 7 8 7 6 5 4 3 2 1      1
      (h)                      (i)

1          1  1 2 3 4 5 6 7 8 7 6 5 4 3 2 1
1 2          2 1  1 2 3 4 5 6 7 7 6 5 4 3 2 1
1 2 3          3 2 1  1 2 3 4 5 6 6 5 4 3 2 1
1 2 3 4          4 3 2 1  1 2 3 4 5 5 4 3 2 1
1 2 3 4 5          5 4 3 2 1  1 2 3 4 4 3 2 1
1 2 3 4 5 6          6 5 4 3 2 1  1 2 3 3 2 1
1 2 3 4 5 6 7 7 6 5 4 3 2 1  1 2          2 1
1 2 3 4 5 6 7 8 7 6 5 4 3 2 1  1          1
      (j)                      (k)

      1
     2 3 2
    3 4 5 4 3
   4 5 6 7 6 5 4
  5 6 7 8 9 8 7 6 5
 6 7 8 9 0 1 0 9 8 7 6
7 8 9 0 1 2 3 2 1 0 9 8 7
8 9 0 1 2 3 4 5 4 3 2 1 0 9 8
      (l)

```

11.5 Print Triangles (nested-loop)

Write a method to print each of the following patterns using nested-loops in a class called **PrintTriangles**. The program shall prompt user for the number of rows. The signatures of the methods are:

```
public static void printXxx(int numRows); // Xxx is the pattern's name
```

```

      1
     1 2 1
    1 2 4 2 1
   1 2 4 8 4 2 1
  1 2 4 8 16 8 4 2 1
 1 2 4 8 16 32 16 8 4 2 1
1 2 4 8 16 32 64 32 16 8 4 2 1
1 2 4 8 16 32 64 128 64 32 16 8 4 2 1
      (a) PowerOf2Triangle

```

```

1          1
1 1          1 1
1 2 1          1 2 1
1 3 3 1          1 3 3 1
1 4 6 4 1          1 4 6 4 1
1 5 10 10 5 1          1 5 10 10 5 1
1 6 15 20 15 6 1          1 6 15 20 15 6 1
(b) PascalTriangle1      (c) PascalTriangle2

```

11.6 Trigonometric Series

Write a method to compute $\sin(x)$ and $\cos(x)$ using the following series expansion, in a class called **TrigonometricSeries**. The signatures of the methods are:

```
public static double sin(double x, int numTerms); // x in radians, NOT degrees
public static double cos(double x, int numTerms);
```

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \dots$$

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \dots$$

Compare the values computed using the series with the JDK methods `Math.sin()`, `Math.cos()` at $x=0$, $\pi/6$, $\pi/4$, $\pi/3$, $\pi/2$ using various numbers of terms.

Hints

Do not use `int` to compute the factorial; as factorial of 13 is outside the `int` range. Avoid generating large numerator and denominator. Use `double` to compute the terms as:

$$\frac{x^n}{n!} = \left(\frac{x}{n}\right) \left(\frac{x}{n-1}\right) \dots \left(\frac{x}{1}\right)$$

11.7 Exponential Series

Write a method to compute e and $\exp(x)$ using the following series expansion, in a class called **ExponentialSeries**. The signatures of the methods are:

```
public static double exp(int numTerms); // x in radians
public static double exp(double x, int numTerms);
```

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \dots$$

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots$$

11.8 Special Series

Write a method to compute the sum of the series in a class called **SpecialSeries**. The signature of the method is:

```
public static double specialSeries(double x, int numTerms);
```

$$x + \frac{1}{2} \times \frac{x^3}{3} + \frac{1 \times 3}{2 \times 4} \times \frac{x^5}{5} + \frac{1 \times 3 \times 5}{2 \times 4 \times 6} \times \frac{x^7}{7} + \frac{1 \times 3 \times 5 \times 7}{2 \times 4 \times 6 \times 8} \times \frac{x^9}{9} + \dots; -1 \leq x \leq 1$$

11.9 FactorialInt (Handling Overflow)

Write a program called **FactorialInt** to list all the factorials that can be expressed as an `int` (i.e., 32-bit signed integer in the range of $[-2147483648, 2147483647]$). Your output shall look like:

The factorial of 1 is 1

The factorial of 2 is 2

...

The factorial of 12 is 479001600

The factorial of 13 is out of range

Hints

The maximum and minimum values of a 32-bit `int` are kept in constants `Integer.MAX_VALUE` and `Integer.MIN_VALUE`, respectively. Try these statements:

```
System.out.println(Integer.MAX_VALUE);
System.out.println(Integer.MIN_VALUE);
```

```
System.out.println(Integer.MAX_VALUE + 1);
```

Take note that in the third statement, Java Runtime does not flag out an overflow error, but silently wraps the number around. Hence, you cannot use $F(n) * (n+1) > \text{Integer.MAX_VALUE}$ to check for overflow. Instead, overflow occurs for $F(n+1)$ if $(\text{Integer.MAX_VALUE} / \text{Factorial}(n)) < (n+1)$, i.e., no more room for the next number.

Try

Modify your program called **FactorialLong** to list all the factorial that can be expressed as a long (64-bit signed integer). The maximum value for long is kept in a constant called `Long.MAX_VALUE`.

11.10 FibonacciInt (Handling Overflow)

Write a program called **FibonacciInt** to list all the Fibonacci numbers, which can be expressed as an int (i.e., 32-bit signed integer in the range of $[-2147483648, 2147483647]$). The output shall look like:

```
F(0) = 1
F(1) = 1
F(2) = 2
...
F(45) = 1836311903
F(46) is out of the range of int
```

Hints

The maximum and minimum values of a 32-bit int are kept in constants `Integer.MAX_VALUE` and `Integer.MIN_VALUE`, respectively. Try these statements:

```
System.out.println(Integer.MAX_VALUE);
System.out.println(Integer.MIN_VALUE);
System.out.println(Integer.MAX_VALUE + 1);
```

Take note that in the third statement, Java Runtime does not flag out an overflow error, but silently wraps the number around. Hence, you cannot use $F(n) = F(n-1) + F(n-2) > \text{Integer.MAX_VALUE}$ to check for overflow. Instead, overflow occurs for $F(n)$ if $\text{Integer.MAX_VALUE} - F(n-1) < F(n-2)$ (i.e., no more room for the next Fibonacci number).

Try

Write a similar program called **TribonacciInt** for Tribonacci numbers.

11.11 Number System Conversion

Write a method call **toRadix()** which converts a positive integer from one radix into another. The method has the following header:

```
public static String toRadix(String in, int inRadix, int outRadix) // The input and output are treated as String.
```

Write a program called **NumberConversion**, which prompts the user for an input string, an input radix, and an output radix, and display the converted number. The output shall look like:

```
Enter a number and radix: A1B2
Enter the input radix: 16
Enter the output radix: 2
"A1B2" in radix 16 is "1010000110110010" in radix 2.
```

11.12 NumberGuess

Write a program called **NumberGuess** to play the number guessing game. The program shall generate a random number between 0 and 99. The player inputs his/her guess, and the program shall response with "Try higher", "Try lower" or "You got it in n trials" accordingly. For example:

```
java NumberGuess
Key in your guess:
50
Try higher
70
Try lower
65
Try lower
61
You got it in 4 trials!
```

Hints

Use `Math.random()` to produce a random number in double between 0.0 (inclusive) and 1.0 (exclusive). To produce an int between 0 and 99, use:

```
final int SECRET_NUMBER = (int)(Math.random()*100); // truncate to int
```

11.13 WordGuess

Write a program called **WordGuess** to guess a word by trying to guess the individual characters. The word to be guessed shall be provided using the command-line argument. Your program shall look like:

```
java WordGuess testing
Key in one character or your guess word: t
Trial 1: t__t__
Key in one character or your guess word: g
Trial 2: t__t__g
Key in one character or your guess word: e
Trial 3: te_t__g
Key in one character or your guess word: testing
Congratulation!
You got in 4 trials
```

Hints

Set up a boolean array (of the length of the word to be guessed) to indicate the positions of the word that have been guessed correctly.

Check the length of the input String to determine whether the player enters a single character or a guessed word. If the player enters a single character, check it against the word to be guessed, and update the boolean array that keeping the result so far.

Try

Try retrieving the word to be guessed from a text file (or a dictionary) randomly.

11.14 DateUtil

Complete the following methods in a class called **DateUtil**:

- `boolean isLeapYear(int year)`: returns true if the given year is a leap year. A year is a leap year if it is divisible by 4 but not by 100, or it is divisible by 400.
- `boolean isValidDate(int year, int month, int day)`: returns true if the given year, month and day constitute a given date. Assume that year is between 1 and 9999, month is between 1 (Jan) to 12 (Dec) and day shall be between 1 and 28|29|30|31 depending on the month and whether it is a leap year.

- `int getDayOfWeek(int year, int month, int day)`: returns the day of the week, where 0 for SUN, 1 for MON, ..., 6 for SAT, for the given date. Assume that the date is valid.
- `String toString(int year, int month, int day)`: prints the given date in the format "xxxday d mmm yyyy", e.g., "Tuesday 14 Feb 2012". Assume that the given date is valid.

Hints

To find the day of the week (Reference: Wiki "Determination of the day of the week"):

1700-	1800-	1900-	2000-	2100-	2200-	2300-	2400-
4	2	0	6	4	2	0	6

1. Based on the first two digit of the year, get the number from the following "century" table.
2. Take note that the entries 4, 2, 0, 6 repeat.
3. Add to the last two digit of the year.
4. Add to "the last two digit of the year divide by 4, truncate the fractional part".
5. Add to the number obtained from the following month table:

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Non-Leap Year	0	3	3	6	1	4	6	2	5	0	3	5
Leap Year	6	2	same as above									

6. Add to the day.
7. The sum modulus 7 gives the day of the week, where 0 for SUN, 1 for MON, ..., 6 for SAT.

For example: 2012, Feb, 17

$(6 + 12 + 12/4 + 2 + 17) \% 7 = 5$ (Fri)

The skeleton of the program is as follows:

```
/* Utilities for Date Manipulation */
public class DateUtil {

    // Month's name – for printing
    public static String[] strMonths
        = { "Jan", "Feb", "Mar", "Apr", "May", "Jun",
            "Jul", "Aug", "Sep", "Oct", "Nov", "Dec" };

    // Number of days in each month (for non-leap years)
    public static int[] daysInMonths
        = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

    // Returns true if the given year is a leap year
    public static boolean isLeapYear(int year) { ..... }

    // Return true if the given year, month, day is a valid date
    // year: 1-9999
    // month: 1(Jan)-12(Dec)
    // day: 1-28|29|30|31. The last day depends on year and month
    public static boolean isValidDate(int year, int month, int day) { ..... }

    // Return the day of the week, 0:Sun, 1:Mon, ..., 6:Sat
    public static int getDayOfWeek(int year, int month, int day) { ..... }

    // Return String "xxxday d mmm yyyy" (e.g., Wednesday 29 Feb 2012)
```



```

public static String printDate(int year, int month, int day) { ..... }

// Test Driver
public static void main(String[] args) {
    System.out.println(isLeapYear(1900)); // false
    System.out.println(isLeapYear(2000)); // true
    System.out.println(isLeapYear(2011)); // false
    System.out.println(isLeapYear(2012)); // true

    System.out.println(isValidDate(2012, 2, 29)); // true
    System.out.println(isValidDate(2011, 2, 29)); // false
    System.out.println(isValidDate(2099, 12, 31)); // true
    System.out.println(isValidDate(2099, 12, 32)); // false

    System.out.println(getDayOfWeek(1982, 4, 24)); // 6:Sat
    System.out.println(getDayOfWeek(2000, 1, 1)); // 6:Sat
    System.out.println(getDayOfWeek(2054, 6, 19)); // 5:Fri
    System.out.println(getDayOfWeek(2012, 2, 17)); // 5:Fri

    System.out.println(toString(2012, 2, 14)); // Tuesday 14 Feb 2012
}
}

```

Notes

You can compare the day obtained with the Java's Calendar class as follows:

```

// Construct a Calendar instance with the given year, month and day
Calendar cal = new GregorianCalendar(year, month - 1, day); // month is 0-based
// Get the day of the week number: 1 (Sunday) to 7 (Saturday)
int dayNumber = cal.get(Calendar.DAY_OF_WEEK);
String[] calendarDays = { "Sunday", "Monday", "Tuesday", "Wednesday",
    "Thursday", "Friday", "Saturday" };

// Print result
System.out.println("It is " + calendarDays[dayNumber - 1]);

```

The calendar we used today is known as Gregorian calendar, which came into effect in October 15, 1582 in some countries and later in other countries. It replaces the Julian calendar. 10 days were removed from the calendar, i.e., October 4, 1582 (Julian) was followed by October 15, 1582 (Gregorian). The only difference between the Gregorian and the Julian calendar is the "leap-year rule". In Julian calendar, every four years is a leap year. In Gregorian calendar, a leap year is a year that is divisible by 4 but not divisible by 100, or it is divisible by 400, i.e., the Gregorian calendar omits century years which are not divisible by 400. Furthermore, Julian calendar considers the first day of the year as march 25th, instead of January 1st.

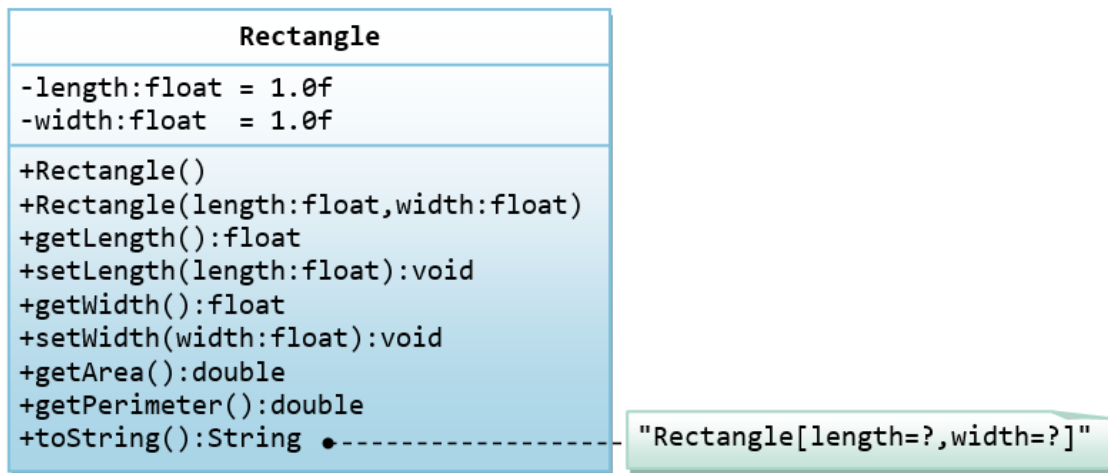
This above algorithm work for Gregorian dates only. It is difficult to modify the above algorithm to handle pre-Gregorian dates. A better algorithm is to find the number of days from a known date.

12. Exercises on Classes and Objects

12.1 The Rectangle Class

A class called Rectangle, which models a rectangle with a length and a width (in float), is designed as shown in the following class diagram. Write the Rectangle class.

Hints:



The expected output is:

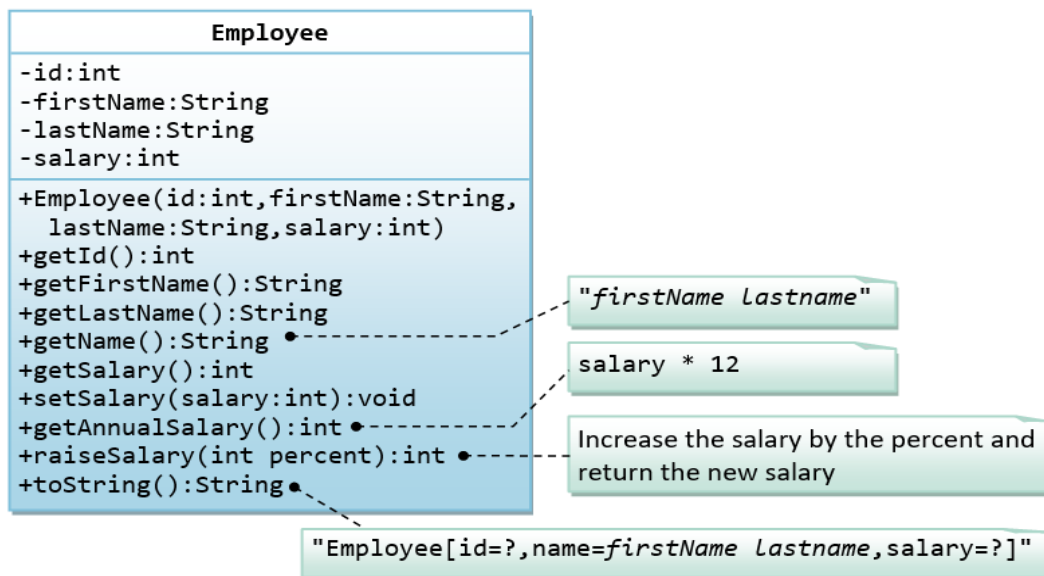
```

Rectangle[length=1.2,width=3.4]
Rectangle[length=1.0,width=1.0]
Rectangle[length=5.6,width=7.8]
length is: 5.6
width is: 7.8
area is: 43.68
perimeter is: 26.80
  
```

12.2 The Employee Class

A class called Employee, which models an employee with an ID, name and salary, is designed as shown in the following class diagram. The method raiseSalary(percent) increases the salary by the given percentage. Write the Employee class.

Hints:



The expected out is:

```

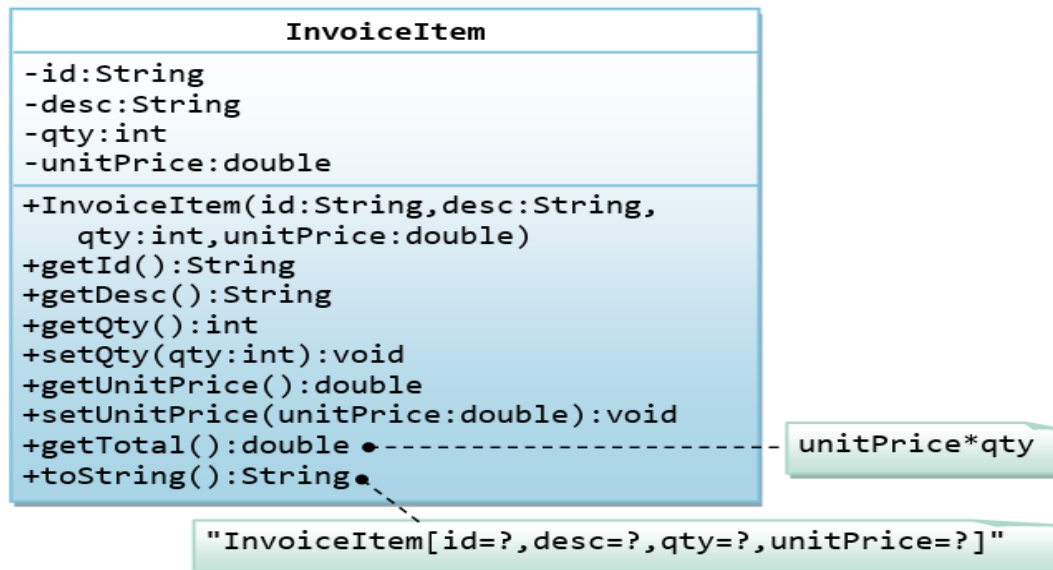
Employee[id=8,name=Peter Tan,salary=2500]
Employee[id=8,name=Peter Tan,salary=999]
id is: 8
firstname is: Peter
lastname is: Tan
salary is: 999
name is: Peter Tan
annual salary is: 11988
1098
  
```

```
Employee[id=8,name=Peter Tan,salary=1098]
```

12.3 The InvoiceItem Class

A class called InvoiceItem, which models an item of an invoice, with ID, description, quantity and unit price, is designed as shown in the following class diagram. Write the InvoiceItem class.

Hints:



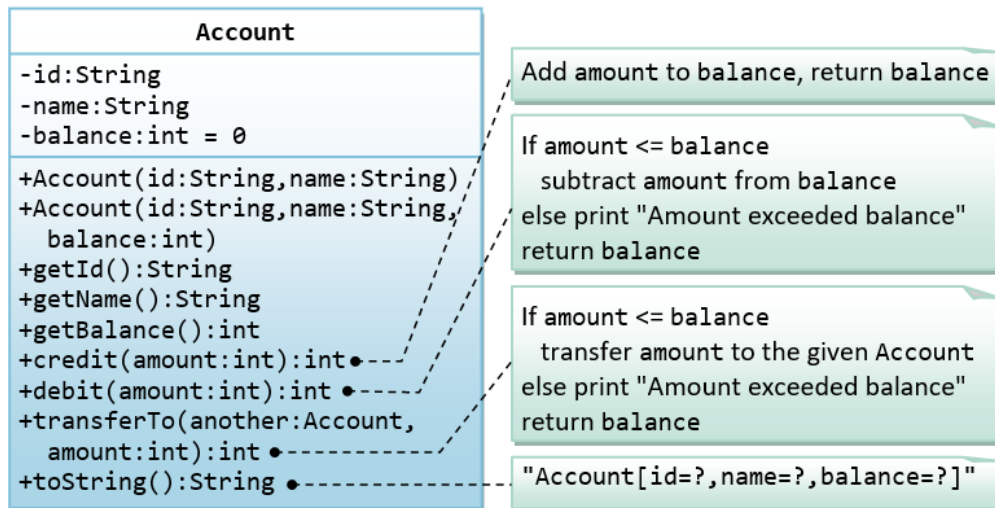
The expected output is:

```
InvoiceItem[id=A101,desc=Pen Red,qty=888,unitPrice=0.08]
InvoiceItem[id=A101,desc=Pen Red,qty=999,unitPrice=0.99]
id is: A101
desc is: Pen Red
qty is: 999
unitPrice is: 0.99
The total is: 989.01
```

12.4 The Account Class

A class called Account, which models a bank account of a customer, is designed as shown in the following class diagram. The methods `credit(amount)` and `debit(amount)` add or subtract the given amount to the balance. The method `transferTo(anotherAccount, amount)` transfers the given amount from this Account to the given anotherAccount. Write the Account class.

Hints:



The expected output is:

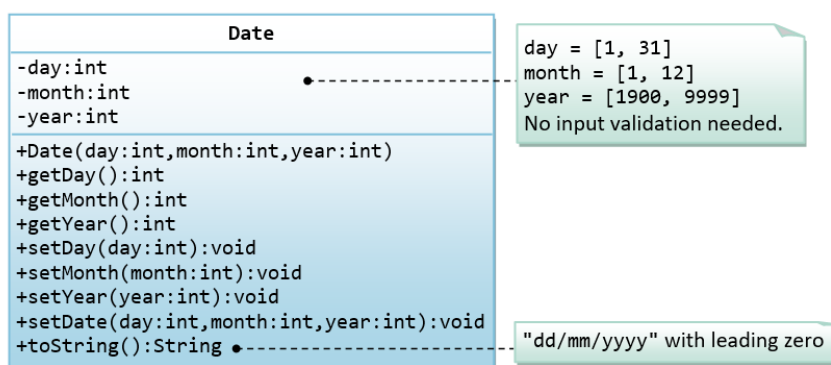
```

Account[id=A101,name=Tan Ah Teck,balance=88]
Account[id=A102,name=Kumar,balance=0]
ID: A101
Name: Tan Ah Teck
Balance: 88
Account[id=A101,name=Tan Ah Teck,balance=188]
Account[id=A101,name=Tan Ah Teck,balance=138]
Amount exceeded balance
Account[id=A101,name=Tan Ah Teck,balance=138]
Account[id=A101,name=Tan Ah Teck,balance=38]
Account[id=A102,name=Kumar,balance=100]
  
```

12.5 The Date Class

A class called Date, which models a calendar date, is designed as shown in the following class diagram. Write the Date class.

Hints:



The expected output is:

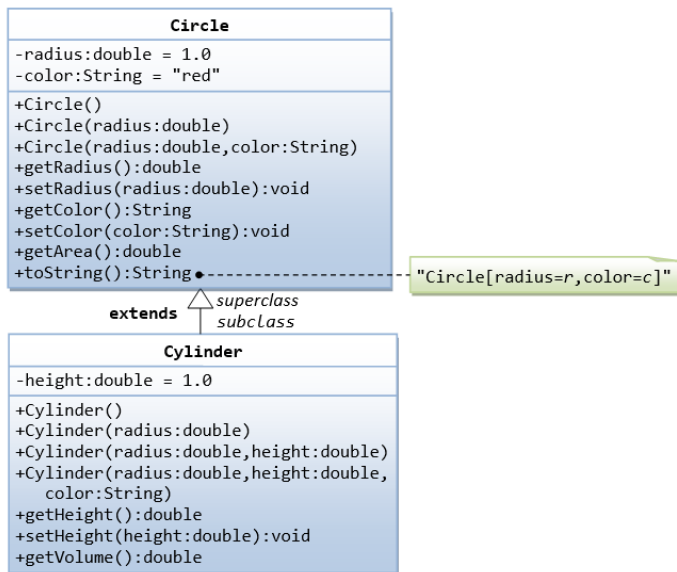
```

01/02/2014
09/12/2099
Month: 12
Day: 9
Year: 2099
03/04/2016
  
```

13. Exercises on Inheritance

13.1 An Introduction to OOP Inheritance - The Circle and Cylinder Classes

This exercise shall guide you through the important concepts in inheritance.



In this exercise, a subclass called Cylinder is derived from the superclass Circle as shown in the class diagram (where an arrow pointing up from the subclass to its superclass). Study how the subclass Cylinder invokes the superclass' constructors (via `super()` and `super(radius)`) and inherits the variables and methods from the superclass Circle.

You can reuse the Circle class that you have created in the previous exercise. Make sure that you keep "Circle.class" in the same directory.

```
public class Cylinder extends Circle { // Save as "Cylinder.java"
    private double height; // private variable

    // Constructor with default color, radius and height
    public Cylinder() {
        super(); // call superclass no-arg constructor Circle()
        height = 1.0;
    }
    // Constructor with default radius, color but given height
    public Cylinder(double height) {
        super(); // call superclass no-arg constructor Circle()
        this.height = height;
    }
    // Constructor with default color, but given radius, height
    public Cylinder(double radius, double height) {
        super(radius); // call superclass constructor Circle(r)
        this.height = height;
    }

    // A public method for retrieving the height
    public double getHeight() {
        return height;
    }

    // A public method for computing the volume of cylinder
    // use superclass method getArea() to get the base area
    public double getVolume() {
        return getArea()*height;
    }
}
```

Method Overriding and "Super": The subclass Cylinder inherits getArea() method from its superclass Circle. Try overriding the getArea() method in the subclass Cylinder to compute the surface area ($=2\pi \times \text{radius} \times \text{height} + 2 \times \text{base-area}$) of the cylinder instead of base area. That is, if getArea() is called by a Circle instance, it returns the area. If getArea() is called by a Cylinder instance, it returns the surface area of the cylinder.

If you override the getArea() in the subclass Cylinder, the getVolume() no longer works. This is because the getVolume() uses the overridden getArea() method found in the same class. (Java runtime will search the superclass only if it cannot locate the method in this class). Fix the getVolume().

Hints: After overriding the getArea() in subclass Cylinder, you can choose to invoke the getArea() of the superclass Circle by calling super.getArea().

Try:

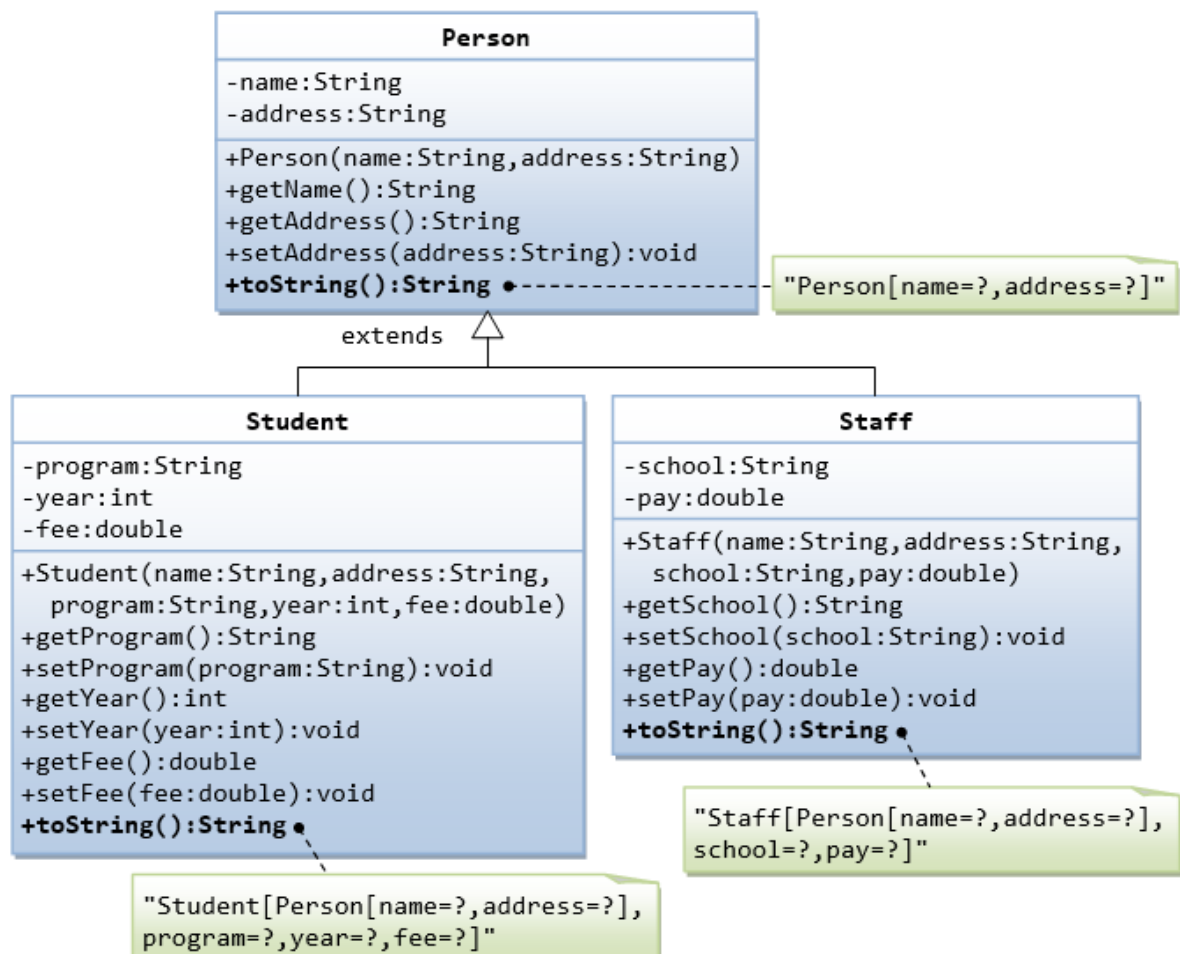
Provide a toString() method to the Cylinder class, which overrides the toString() inherited from the superclass Circle, e.g.,

```
@Override
public String toString() {    // in Cylinder class
    return "Cylinder: subclass of " + super.toString() // use Circle's toString()
        + " height=" + height;
}
```

Try out the toString() method in TestCylinder.

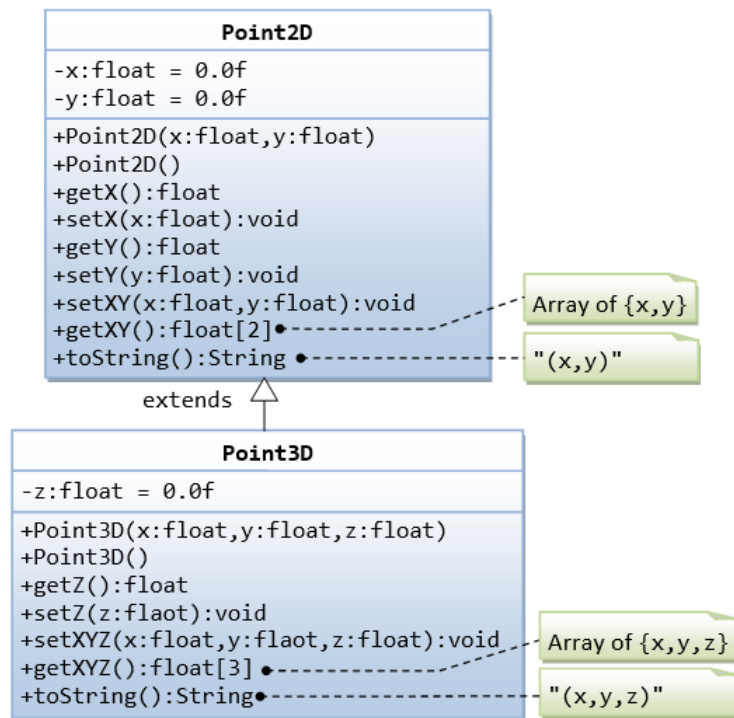
13.2 Superclass Person and its subclasses

Write the classes as shown in the following class diagram. Mark all the overridden methods with annotation @Override.



13.3 Point2D and Point3D

Write the classes as shown in the following class diagram. Mark all the overridden methods with annotation `@Override`.



Hints:

1. You cannot assign floating-point literal say 1.1 (which is a double) to a float variable, you need to add a suffix f, e.g. 0.0f, 1.1f.
2. The instance variables x and y are private in Point2D and cannot be accessed directly in the subclass Point3D. You need to access via the public getters and setters. For example,

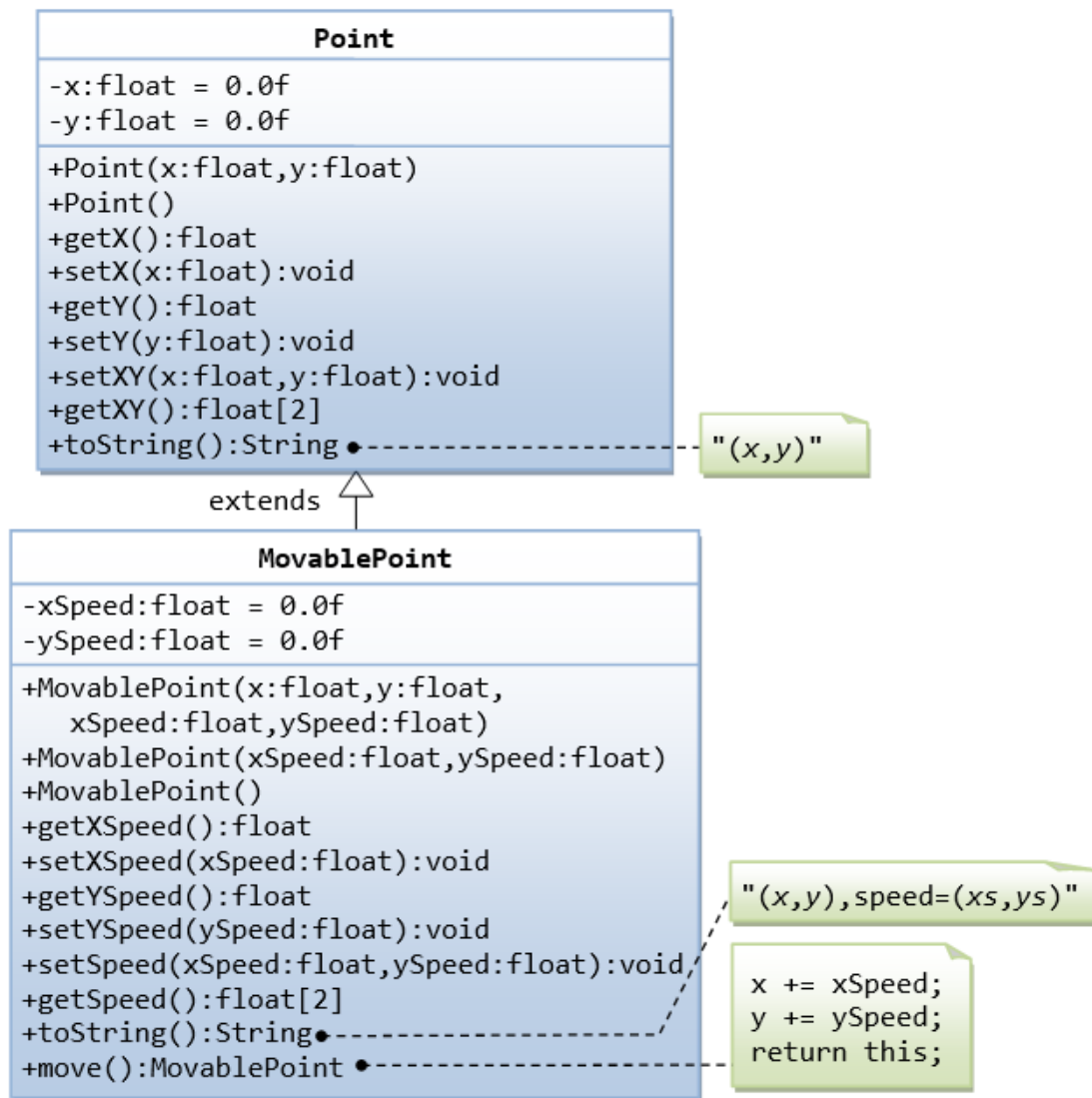
```
public void setXYZ(float x, float y, float z) {
    setX(x); // or super.setX(x), use setter in superclass
    setY(y);
    this.z = z;
}
```

3. The method getXY() shall return a float array:

```
public float[] getXY() {
    float[] result = new float[2]; // construct an array of 2 elements
    result[0] = ...
    result[1] = ...
    return result; // return the array
}
```

13.4 Point and MovablePoint

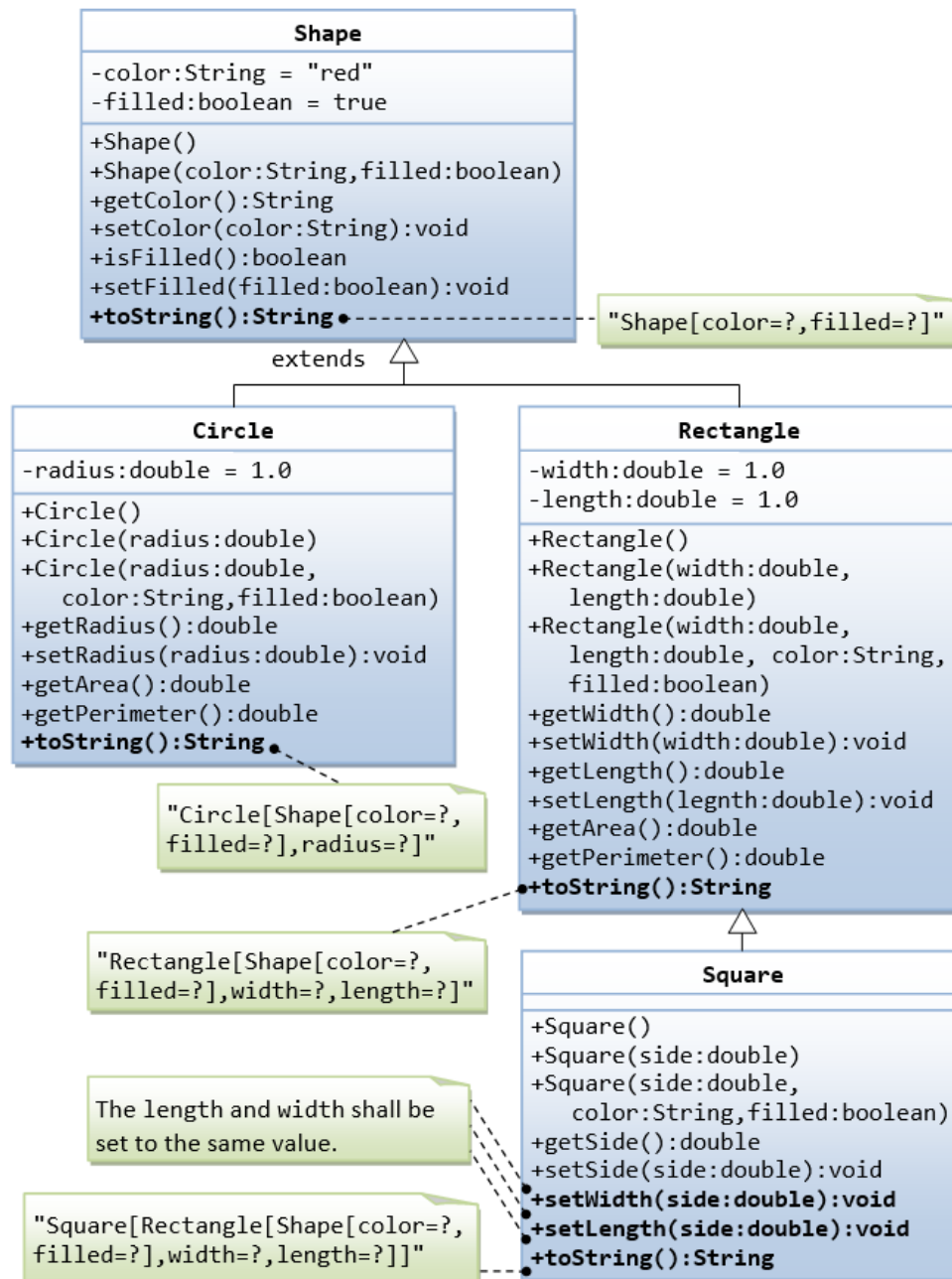
Write the classes as shown in the following class diagram. Mark all the overridden methods with annotation `@Override`.



Hints

1. You cannot assign floating-point literal say 1.1 (which is a double) to a float variable, you need to add a suffix f, e.g. 0.0f, 1.1f.
2. The instance variables x and y are private in Point and cannot be accessed directly in the subclass MovablePoint. You need to access via the public getters and setters. For example, you cannot write `x += xSpeed`, you need to write `setX(getX() + xSpeed)`.

13.5 Superclass **Shape** and its subclasses **Circle**, **Rectangle** and **Square**



Write a superclass called Shape (as shown in the class diagram)

Write a test program to test all the methods defined in Shape.

Write two subclasses of Shape called Circle and Rectangle, as shown in the class diagram.

Write a class called Square, as a subclass of Rectangle. Convince yourself that Square can be modeled as a subclass of Rectangle. Square has no instance variable, but inherits the instance variables width and length from its superclass Rectangle.

- Provide the appropriate constructors (as shown in the class diagram).

Hints:

```

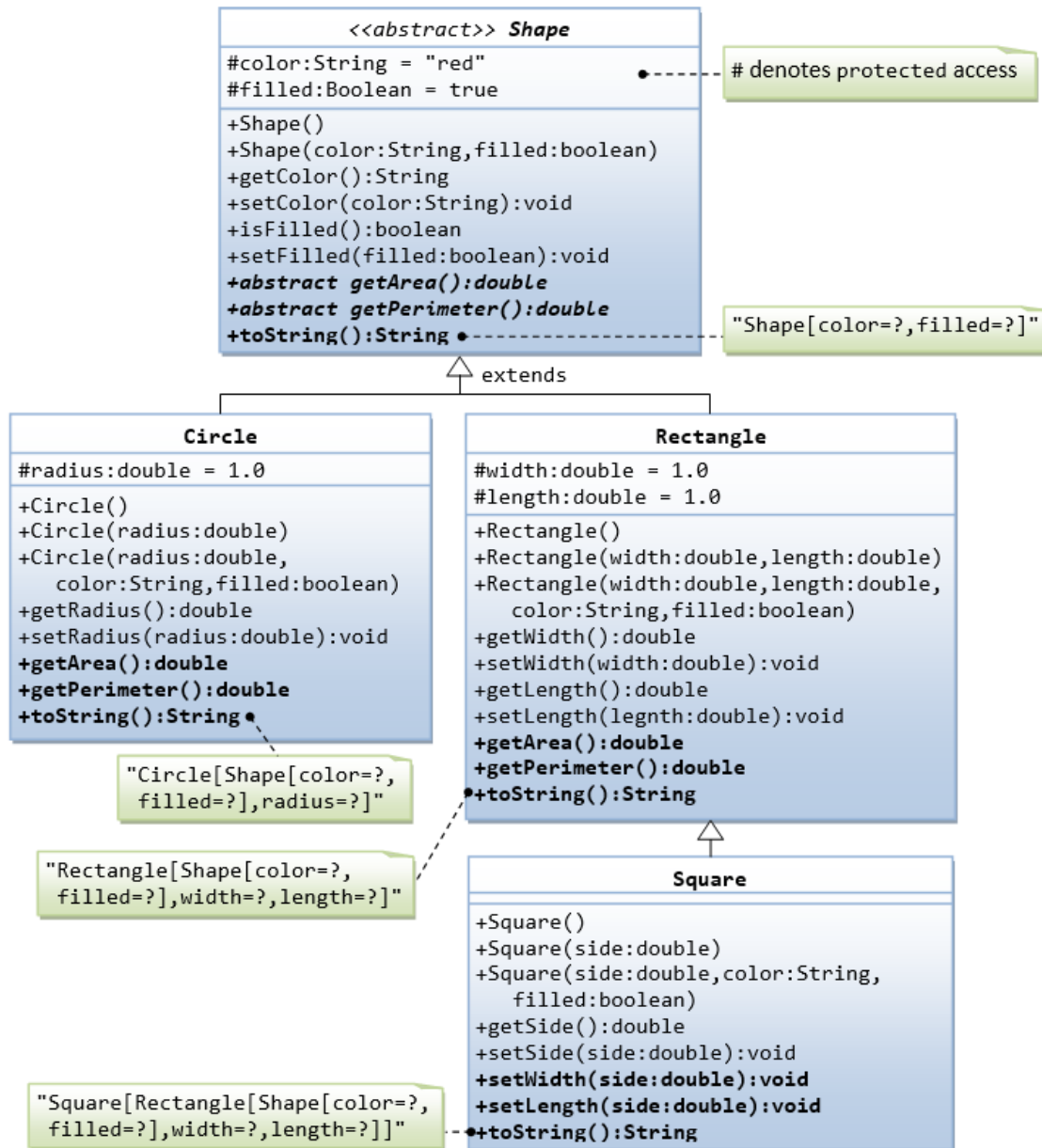
public Square(double side) {
    super(side, side); // Call superclass Rectangle(double, double)
}

```

- Override the `toString()` method to return "A Square with side=xxx, which is a subclass of yyy", where yyy is the output of the `toString()` method from the superclass.
- Do you need to override the `getArea()` and `getPerimeter()`? Try them out.
- Override the `setLength()` and `setWidth()` to change both the width and length, so as to maintain the square geometry.

14.1 Ex: Abstract Superclass **Shape** and Its Concrete Subclasses

Rewrite the superclass Shape and its subclasses Circle, Rectangle and Square, as shown in the class diagram. Shape is an abstract class containing 2 abstract methods: `getArea()` and `getPerimeter()`, where its concrete subclasses must provide its implementation. All instance variables shall have protected access, i.e., accessible by its subclasses and classes in the same package. Mark all the overridden methods with annotation `@Override`.



In this exercise, Shape shall be defined as an abstract class, which contains:

- Two protected instance variables **color(String)** and **filled(boolean)**. The protected variables can be accessed by its subclasses and classes in the same package. They are denoted with a '#' sign in the class diagram.
- Getter and setter for all the instance variables, and `toString()`.
- Two abstract methods `getArea()` and `getPerimeter()` (shown in italics in the class diagram).
- Subclasses Circle and Rectangle shall override the abstract methods `getArea()` and `getPerimeter()` and provide the proper implementation. They also override the `toString()`.

Write a test class to test these statements involving polymorphism and explain the outputs. Some statements may trigger compilation errors. Explain the errors, if any.

```

Shape s1 = new Circle(5.5, "red", false); // Upcast Circle to Shape
System.out.println(s1);                // which version?
System.out.println(s1.getArea());        // which version?
System.out.println(s1.getPerimeter());   // which version?
System.out.println(s1.getColor());
System.out.println(s1.isFilled());
System.out.println(s1.getRadius());

Circle c1 = (Circle)s1;                 // Downcast back to Circle
System.out.println(c1);
System.out.println(c1.getArea());
System.out.println(c1.getPerimeter());
System.out.println(c1.getColor());
System.out.println(c1.isFilled());
System.out.println(c1.getRadius());

Shape s2 = new Shape();

Shape s3 = new Rectangle(1.0, 2.0, "red", false); // Upcast
System.out.println(s3);
System.out.println(s3.getArea());
System.out.println(s3.getPerimeter());
System.out.println(s3.getColor());
System.out.println(s3.getLength());

Rectangle r1 = (Rectangle)s3; // downcast
System.out.println(r1);
System.out.println(r1.getArea());
System.out.println(r1.getColor());
System.out.println(r1.getLength());

Shape s4 = new Square(6.6); // Upcast
System.out.println(s4);
System.out.println(s4.getArea());
System.out.println(s4.getColor());
System.out.println(s4.getSide());

// Take note that we downcast Shape s4 to Rectangle,
// which is a superclass of Square, instead of Square
Rectangle r2 = (Rectangle)s4;
System.out.println(r2);
System.out.println(r2.getArea());
System.out.println(r2.getColor());
System.out.println(r2.getSide());
System.out.println(r2.getLength());

// Downcast Rectangle r2 to Square
Square sq1 = (Square)r2;
System.out.println(sq1);
System.out.println(sq1.getArea());
System.out.println(sq1.getColor());
System.out.println(sq1.getSide());
System.out.println(sq1.getLength());

```

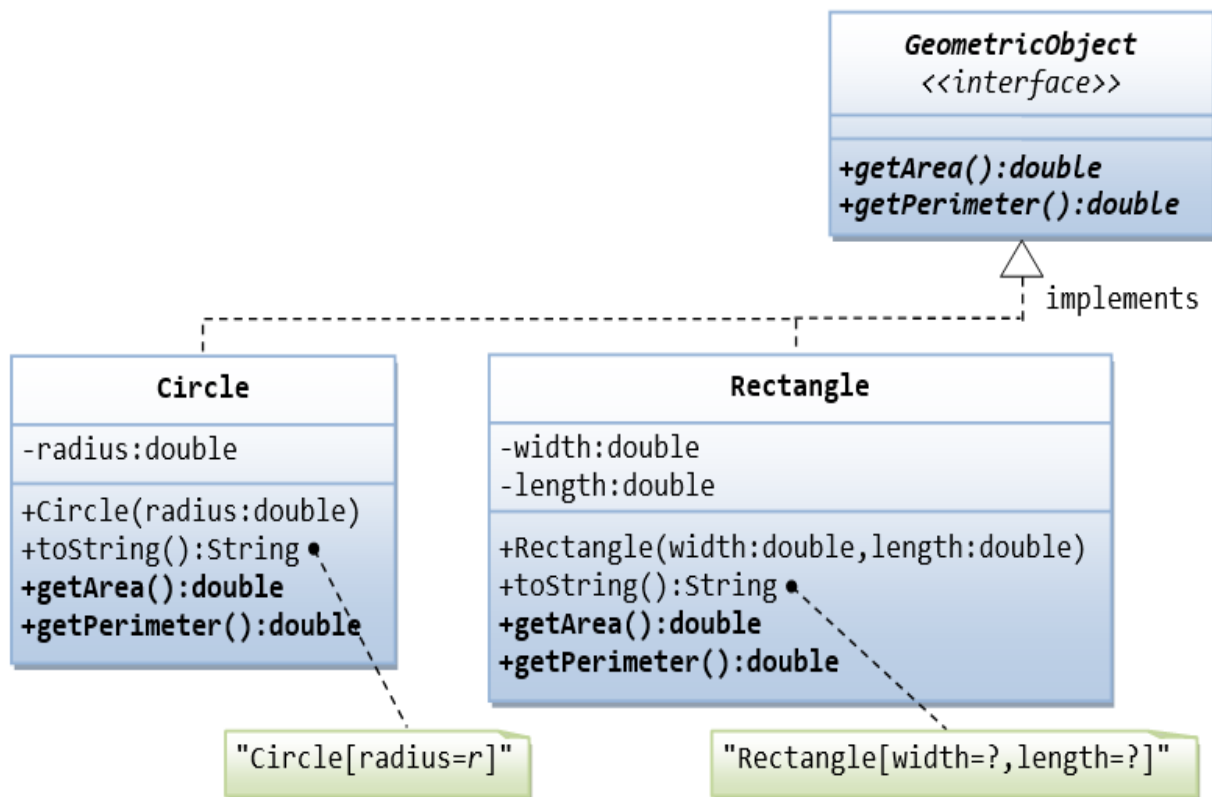
Try:

Explain the usage of the abstract method and abstract class?

14.2 GeometricObject Interface and its Implementation

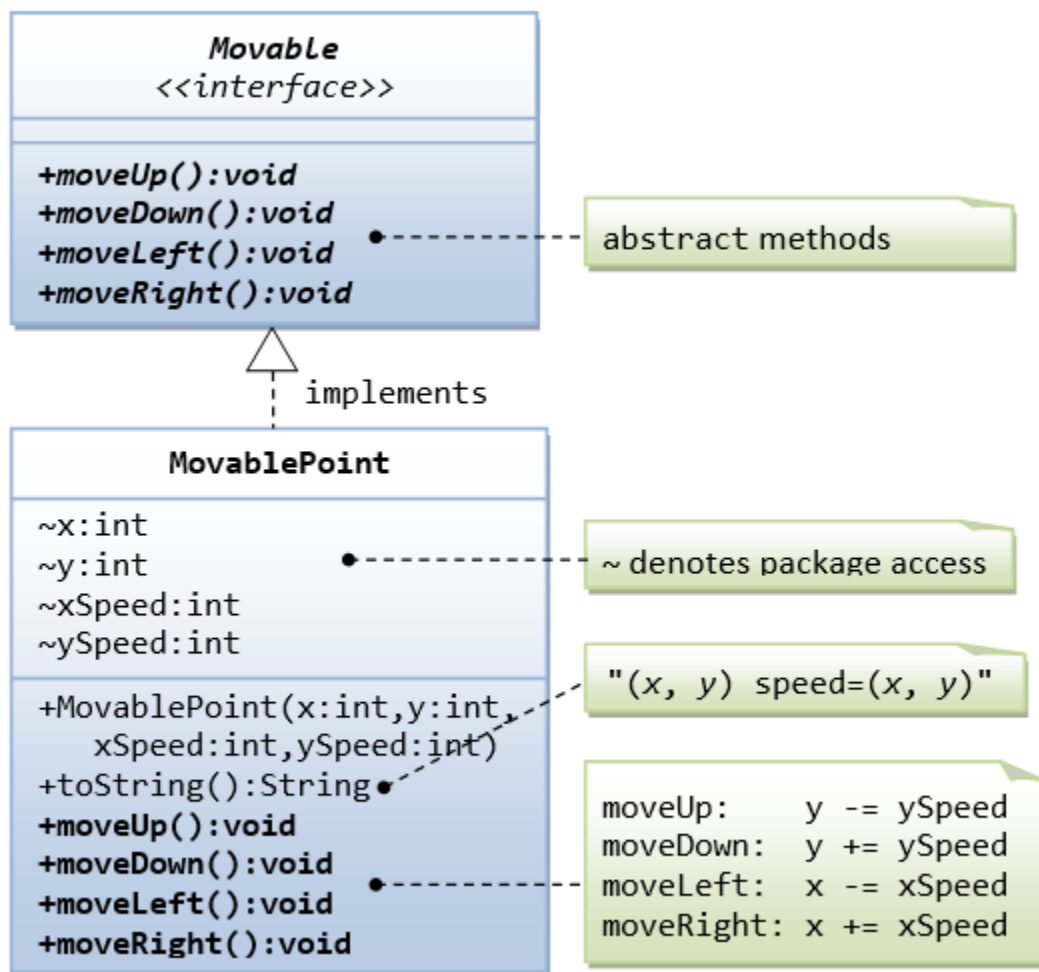
Classes Circle and Rectangle

Write an interface called GeometricObject, which contains 2 abstract methods: `getArea()` and `getPerimeter()`, as shown in the class diagram. Also write an implementation class called Circle. Mark all the overridden methods with annotation `@Override`.



14.3 Ex: Movable Interface and its Implementation MovablePoint Class

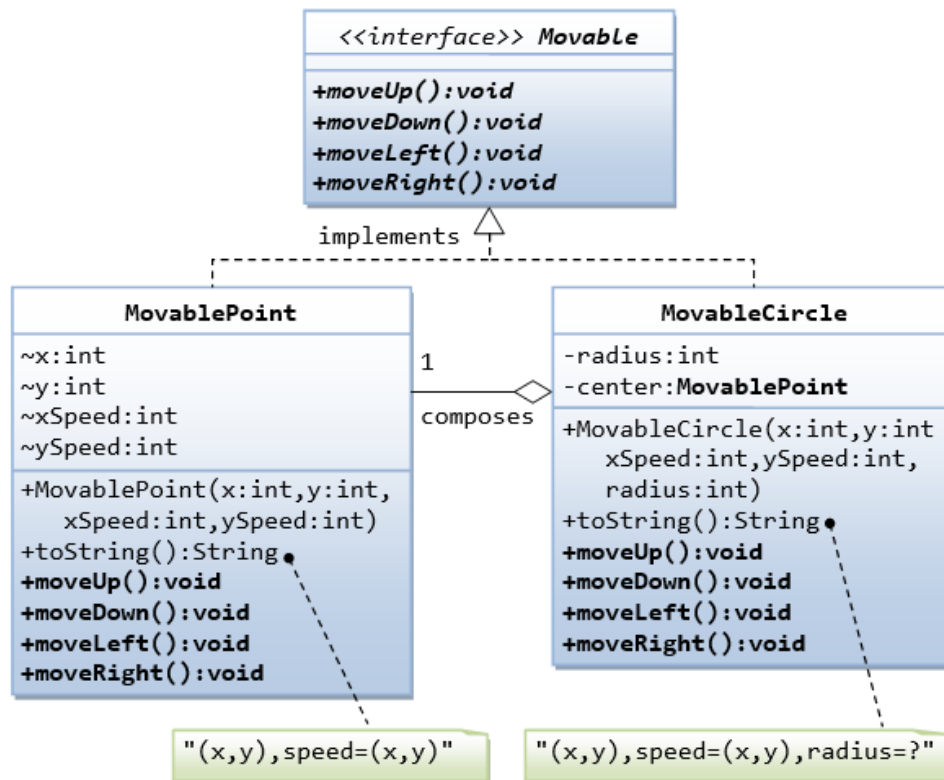
Write an interface called Movable, which contains 4 abstract methods moveUp(), moveDown(), moveLeft() and moveRight(), as shown in the class diagram. Also write an implementation class called MovablePoint. Mark all the overridden methods with annotation @Override.



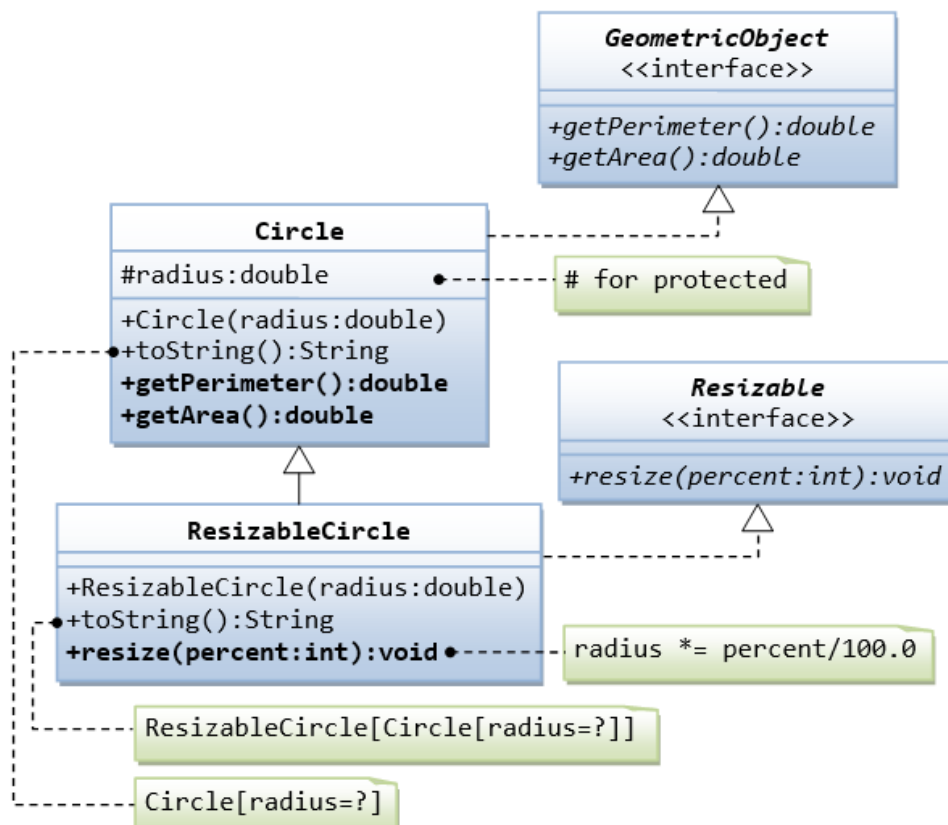
14.4 Movable Interface and Classes MovablePoint and MovableCircle

Write an interface called Movable, which contains 4 abstract methods moveUp(), moveDown(), moveLeft() and moveRight(), as shown in the class diagram.

Write the implementation classes called MovablePoint and MovableCircle. Mark all the overridden methods with annotation @Override.



14.5 Interfaces **Resizable** and **GeometricObject**



Write the interface called **GeometricObject**, which declares two abstract methods: `getPerimeter()` and `getArea()`, as specified in the class diagram.

Hints:

```
public interface GeometricObject {
    public double getPerimeter();
    .....
}
```

Write the implementation class Circle, with a protected variable radius, which implements the interface GeometricObject.

Hints:

```
public class Circle implements GeometricObject {
    // Private variable
    .....

    // Constructor
    .....

    // Implement methods defined in the interface GeometricObject
    @Override
    public double getPerimeter() { ..... }

    .....
}
```

Write a test program called TestCircle to test the methods defined in Circle.

The class ResizableCircle is defined as a subclass of the class Circle, which also implements an interface called Resizable, as shown in class diagram. The interface Resizable declares an abstract method resize(), which modifies the dimension (such as radius) by the given percentage. Write the interface Resizable and the class ResizableCircle.

Hints:

```
public interface Resizable {
    public double resize(...);
}
```

```
public class ResizableCircle extends Circle implements Resizable {

    // Constructor
    public ResizableCircle(double radius) {
        super(...);
    }

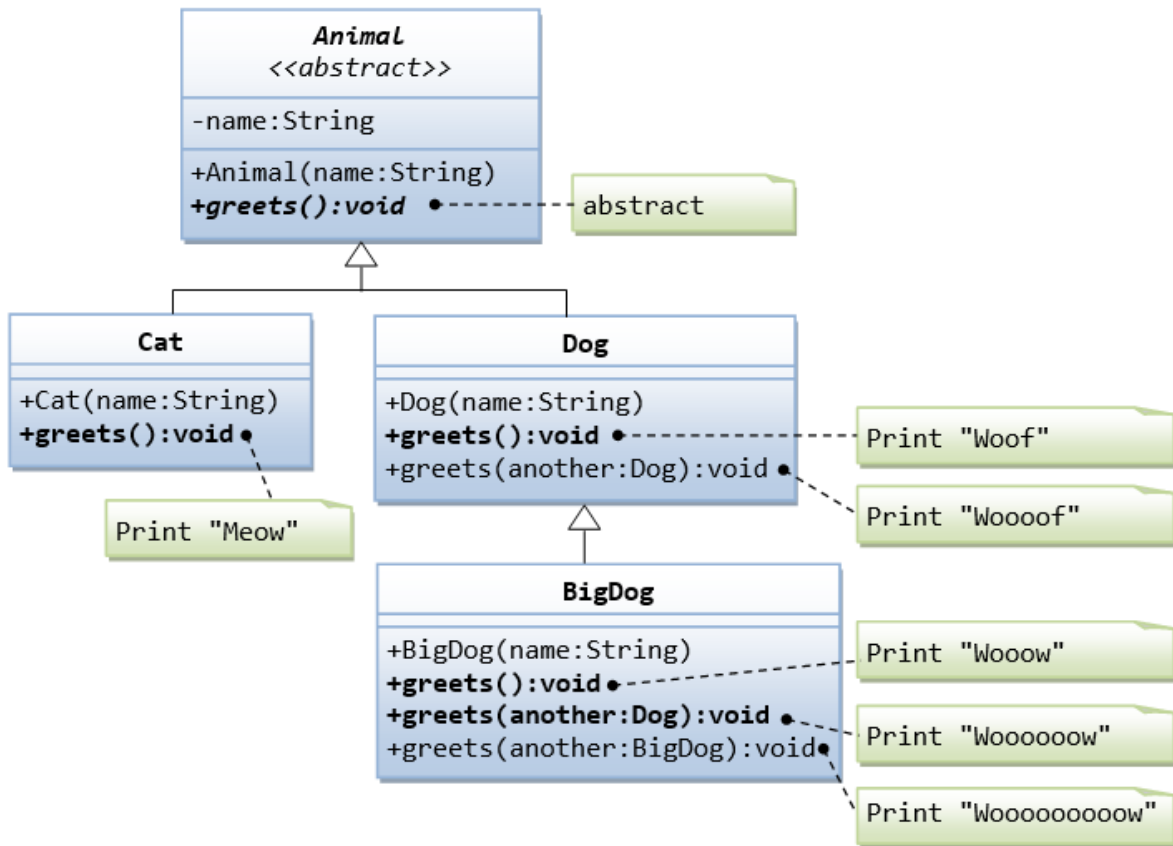
    // Implement methods defined in the interface Resizable
    @Override
    public double resize(int percent) { ..... }
}
```

Try:

Write a test program called TestResizableCircle to test the methods defined in ResizableCircle.

14.6 Abstract Superclass **Animal** and its Implementation Subclasses

Write the codes for all the classes shown in the class diagram. Mark all the overridden methods with annotation @Override.



14.7 Another View of Abstract Superclass **Animal** and its Implementation Subclasses

Examine the following codes and draw the class diagram.

```

abstract public class Animal {
    abstract public void greeting();
}

```

```

public class Cat extends Animal {
    @Override
    public void greeting() {
        System.out.println("Meow!");
    }
}

public class Dog extends Animal {
    @Override
    public void greeting() {
        System.out.println("Woof!");
    }

    public void greeting(Dog another) {
        System.out.println("Woowoooooof!");
    }
}

```

```

public class BigDog extends Dog {
    @Override
    public void greeting() {
        System.out.println("Woow!");
    }

    @Override
    public void greeting(Dog another) {

```



```

        System.out.println("Woooooowwww!");
    }
}

```

Try:

Explain the outputs (or error) for the following test program.

```

public class TestAnimal {
    public static void main(String[] args) {
        // Using the subclasses
        Cat cat1 = new Cat();
        cat1.greeting();
        Dog dog1 = new Dog();
        dog1.greeting();
        BigDog bigDog1 = new BigDog();
        bigDog1.greeting();

        // Using Polymorphism
        Animal animal1 = new Cat();
        animal1.greeting();
        Animal animal2 = new Dog();
        animal2.greeting();
        Animal animal3 = new BigDog();
        animal3.greeting();
        Animal animal4 = new Animal();

        // Downcast
        Dog dog2 = (Dog)animal2;
        BigDog bigDog2 = (BigDog)animal3;
        Dog dog3 = (Dog)animal3;
        Cat cat2 = (Cat)animal2;
        dog2.greeting(dog3);
        dog3.greeting(dog2);
        dog2.greeting(bigDog2);
        bigDog2.greeting(dog2);
        bigDog2.greeting(bigDog1);
    }
}

```

15. Final Notes

The only way to learn programming is program, program and program on challenging problems. The problems in this tutorial are certainly NOT challenging. There are tens of thousands of challenging problems available – used in training for various programming contests (such as International Collegiate Programming Contest (ICPC), International Olympiad in Informatics (IOI)). Check out these sites:

- The ACM - ICPC International collegiate programming contest (<https://icpc.global/>)
- The Topcoder Open (TCO) annual programming and design contest (<https://www.topcoder.com/>)
- Universidad de Valladolid's online judge (<https://uva.onlinejudge.org/>).
- Peking University's online judge (<http://poj.org/>).
- USA Computing Olympiad (USACO) Training Program @ <http://train.usaco.org/usacogate>.
- Google's coding competitions (<https://codingcompetitions.withgoogle.com/codejam>, <https://codingcompetitions.withgoogle.com/hashcode>)
- The ICFP programming contest (<https://www.icfpconference.org/>)
- BME International 24-hours programming contest (<https://www.challenge24.org/>)
- The International Obfuscated C Code Contest (<https://www0.us.ioccc.org/main.html>)
- Internet Problem Solving Contest (<https://ipsc.ksp.sk/>)
- Microsoft Imagine Cup (<https://imaginecup.microsoft.com/en-us>)
- Hewlett Packard Enterprise (HPE) Codewars (<https://hpecodewars.org/>)
- OpenChallenge (<https://www.openchallenge.org/>)

Coding Contests Scores

Students must solve problems and attain scores in the following coding contests:

	Name of the contest	Minimum number of problems to solve	Required score
•	CodeChef	20	200
•	Leetcode	20	200
•	GeeksforGeeks	20	200
•	SPOJ	5	50
•	InterviewBit	10	1000
•	Hackerrank	25	250
•	Codeforces	10	100
•	BuildIT	50	500
	Total score need to obtain		2500

Student must have any one of the following certifications:

- HackerRank – Java Basic Skills Certification
- Oracle Certified Associate Java Programmer OCAJP
- CodeChef - Learn Java Certification
- NPTEL – Programming in Java
- NPTEL – Data Structures and Algorithms in Java

V. TEXT BOOKS:

1. Farrell, Joyce. Java Programming, Cengage Learning B S Publishers, 8th Edition, 2020
2. Schildt, Herbert. Java: The Complete Reference 11th Edition, McGraw-Hill Education, 2018.

VI. REFERENCE BOOKS:

1. Deitel, Paul and Deitel, Harvey. Java: How to Program, Pearson, 11th Edition, 2018.
2. Evans, Benjamin J. and Flanagan, David. Java in a Nutshell, O'Reilly Media, 7th Edition, 2018.
3. Bloch, Joshua. Effective Java, Addison-Wesley Professional, 3rd Edition, 2017.
4. Sierra, Kathy and Bates, Bert. Head First Java, O'Reilly Media, 2nd Edition, 2005

VII. ELECTRONICS RESOURCES:

1. <https://docs.oracle.com/en/java/>
2. <https://www.geeksforgeeks.org/java>
3. <https://www.tutorialspoint.com/java/index.htm>
4. <https://www.coursera.org/courses?query=java>

VIII. MATERIALS ONLINE;

1. Syllabus
2. Lab manual



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

ENGINEERING WORKSHOP								
I Semester: AE / CE / ME								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AMED01	Foundation	L	T	P	C	CIA	SEE	Total
		0	1	2	2	40	60	100
Contact Classes: Nil	Tutorial Classes: 15	Practical Classes: 30			Total Classes:45			
Prerequisite: There is no prerequisite for this course.								

I. COURSE OVERVIEW:

This course provides the opportunity to become confident with new tools, equipment, and techniques for creating physical objects and mechanisms with a variety of materials. The students will learn principles of contemporary trends in manufacturing processes, such as CNC machining and 3D printing, as well as gain practical experience in carpentry, fitting, and welding. Skills learned in the course enable the students to learn about the design process in digital manufacturing used in various industrial applications.

II. COURSES OBJECTIVES:

The students will try to learn

- I. The basics and hands-on practice of carpentry, fitting, and welding
- II. The impart knowledge and skill to use tools, equipment, measuring instruments, and modern techniques.
- III. The concepts apply to the manufacturing processes of casting, moulding and forging.
- IV. The basic machining operations by CNC lathe, CNC milling, and 3D printing machine.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- | | |
|------|--|
| CO 1 | Select appropriate tools, work material and measuring instruments useful for carpentry, fitting, and welding. |
| CO 2 | Use flat sheets for sheet metal and intricate shapes made from mild steel for Black smithy. |
| CO 3 | Choose appropriate components and tools to prepare pipe fitting and joints of specific shapes and sizes. |
| CO 4 | Experiment with the moulding techniques for producing cast components in complex shapes using different patterns. |
| CO 5 | Execute hard soldering techniques to join similar and dissimilar materials used in industries. |
| CO 6 | Demonstrate appropriate equipment and methods for various machining processes used in CNC machines and 3D printing for manufacturing industries. |

IV. COURSE CONTENT:

EXERCISES IN ENGINEERING WORKSHOP

Note: All dimensions are in mm in experiments.

Getting started experiments

Introduction

Engineering workshop provides both tools and equipments (or machinery) that are required for the manufacture of the goods. Students are familiarized with basic workshop practice like Wood working, Sheet metal, metal joining processes, manufacturing processes etc. and required to identify, operate and control various machines, tools and equipments.

Safety

Safety is a vital issue in all workplaces. Before using any equipment and machines or attempt practical work in a workshop everyone must understand basic safety rules. These rules will help keep all safe in the workshop.

Safety Rules:

- Always listen carefully to the teacher and follow instructions.
- When learning how to use a machine, listen very carefully to all the instructions given by the faculty / instructor. Ask questions, especially if you do not fully understand.
- Always wear an apron as it will protect your clothes and holds loose clothing such as ties in place.
- Bags should not be brought into a workshop as people can trip over them.
- Do not use a machine if you have not been shown how to operate it safely by the faculty / instructors
- Know where the emergency stop buttons are positioned in the workshop. If you see an accident at the other side of the workshop you can use the emergency stop button to turn off all electrical power to machines.
- Wherever required, wear protective equipment, such as goggles, safety glasses, masks, gloves, hair nets, etc.
- Always be patient, never rush in the workshop.
- Always use a guard when working on a machine.
- Keep hands away from moving/rotating machinery.
- Use hand tools carefully, keeping both hands behind the cutting edge.
- Report any UNSAFE condition or acts to instructor.
- Report any damage to machines/equipment as this could cause an accident.
- Keep your work area clean.

DO's

- Students must always wear uniform and shoes before entering the lab.
- Proper code of conduct and ethics must be followed in the lab.
- Note down the specifications/drawings before working on the preparation of models.
- Receive the tools and materials required for preparation of models with signing in register.
- Properly fix hacksaw blade in frame with help of instructor.
- Use of safety goggles / face shield during welding.
- Do the models under the supervision/guidance of a faculty/ lab instructor only.
- Keep the sufficient distance from other students while preparing models.
- In case of fire use fire extinguisher/throw the sand provided in the lab.
- In case of any physical injuries or emergencies use first aid box provided.

DONT's

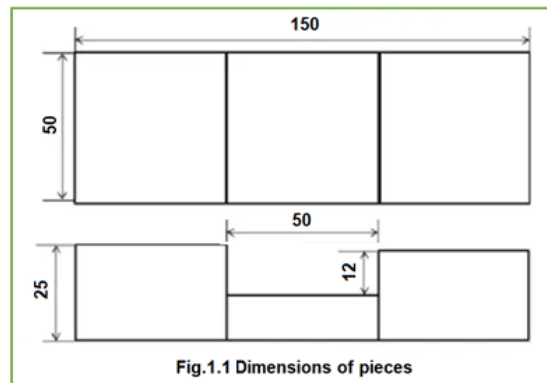
- Do not touch electrical circuits of welding machine.
- Be cautious while fixing hacksaw blade in frame, that may cause injuries to hand.
- Don't touch /operate power tools without aid from instructors.
- Don't gather while preparing models, that may hurt other with tools.
- Don't unlock snip/sheet metal cutter lock, without use.

1. Introduction to carpentry

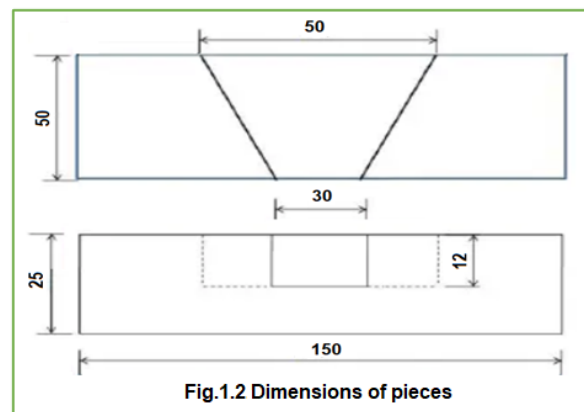
Carpentry is the process of shaping wood, using hand tools. The products produced are used in building construction, such as doors and windows, furniture manufacturing, patterns for moulding in foundries, etc. Carpentry work mainly involves the joining together of wooden pieces and finishing the surfaces after shaping them. Hence, the term joining is also used commonly for carpentry. A student studying the fundamentals of wood working has to know about timber and other carpentry materials, wood working tools, carpentry operations and the method of making common types of joints.

1.1. Experiments on carpentry

1. Preparation of the cross-half lap joint as shown in Fig. 1.1

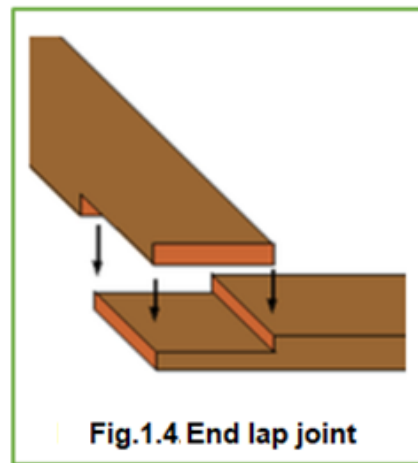


2. Preparation of the dove tail joint as depicted in Fig.1.2



Try

1. Mortise and tenon joint preparation as illustrated in Fig.1.3 with dimensions of width = 50 mm and tenon thickness = 10 mm.
2. End lap joint preparation as illustrated in Fig. 1.4. The end lap projection dimensions to be taken into consideration are width = 50 mm and thickness = 15 mm.

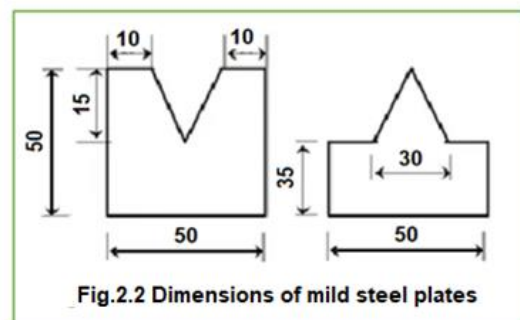
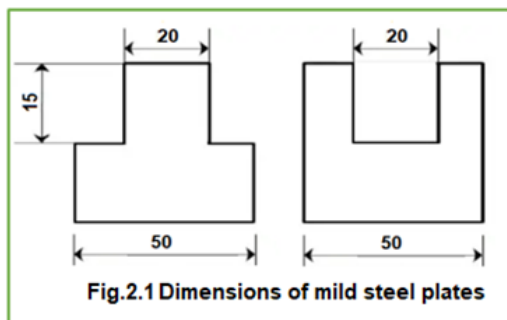


2. Introduction to fitting

The term fitting, is related to assembly of parts, after bringing the dimension or shape to the required size or form, in order to secure the necessary fit. The operations required for the same are usually carried out on a work bench, hence the term bench work is also added with the name fitting. The bench work and fitting play an important role in engineering. Although in today's industries most of the work is done by automatic machines which produces the jobs with good accuracy but still it(job) requires some hand operations called fitting operations.

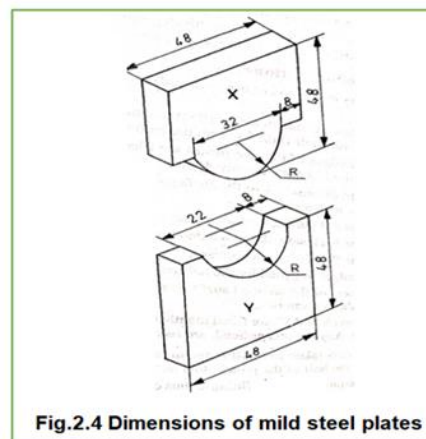
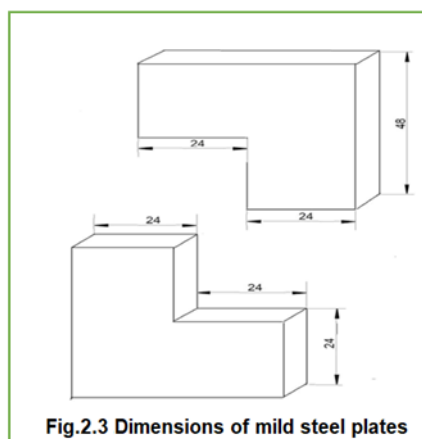
2.1. Experiments on fitting

1. Making of a square fitting using mild steel plates of the specified size, as shown in Fig. 2.1
2. Making of a V-fit according to the size of the provided mild steel plates, as shown in Fig. 2.2



Try

1. Straight fitting of mild steel plates to the specified sizes, as shown in Fig. 2.3
2. Making of semicircular fit with mild steel plates of the specified size, as depicted in Fig. 2.4



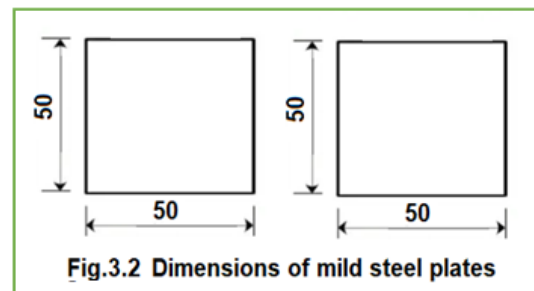
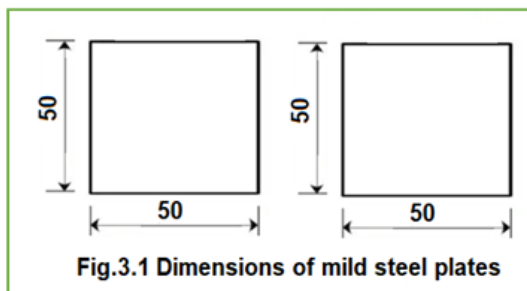
3. Introduction to welding

Welding is a process for joining two similar or dissimilar metals by fusion. It joins different metals/alloys, with or without the application of pressure and with or without the use of filler metal. The fusion of metal takes place by means of heat. The heat may be generated either from combustion of gases, electric arc, electric resistance or by chemical reaction. Welding provides a permanent joint but it normally affects the metallurgy of the components. It is therefore usually accompanied by post weld heat treatment for most of the critical components. The welding is widely used as a fabrication and repairing process in industries. Some of the typical applications of welding include the fabrication of ships, pressure vessels, automobile bodies, off-shore platform, bridges, welded pipes, sealing of nuclear fuel and explosives, etc.

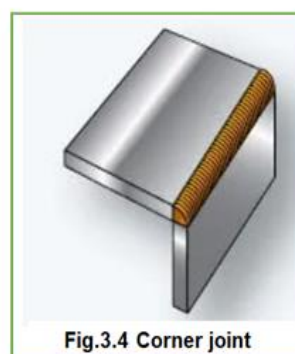
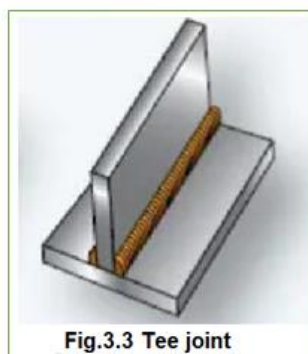
3.1. Experiments and demonstration on different welding techniques

1. Creating the lap joint in accordance with the mild steel plates given, as shown in Fig .3.1
2. Making the butt joint as depicted in Fig. 3.2 using the mild steel plates as are offered.

Try



1. Construction of the tee joint using the mild steel plates provided, as shown in Fig. 3.3
2. As illustrated in Fig. 3.4, creating the corner (L) joint using the provided mild steel plates.



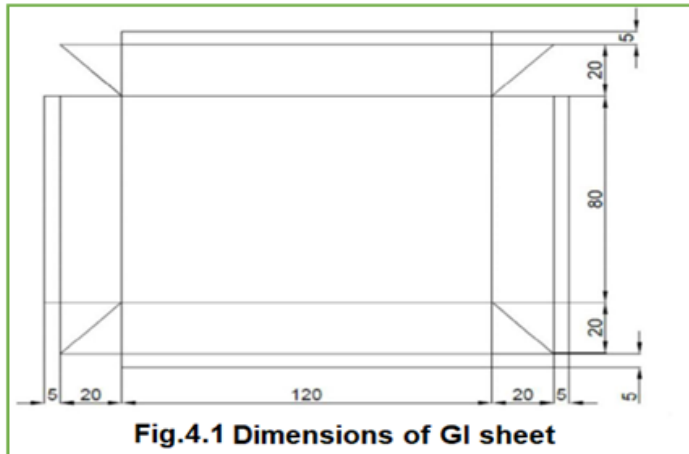
4. sheet metal

Introduction to

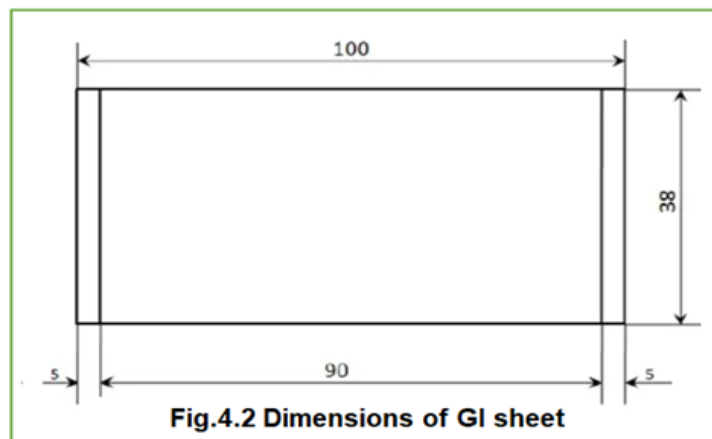
Sheet metal work has its own significance in the engineering work. Many products, which fulfill the household needs, decoration work and various engineering articles, are produced from sheet metals. Common examples of sheet metal work are hoopers, canisters, guards, covers, pipes, hoods, funnels, bends, boxes etc. Such articles are found less expensive, lighter in weight and in some cases sheet metal products replace the use of castings or forgings.

4.1. Experiments on sheet metal forming

1. Create the rectangular tray as depicted in Fig. 4.1.

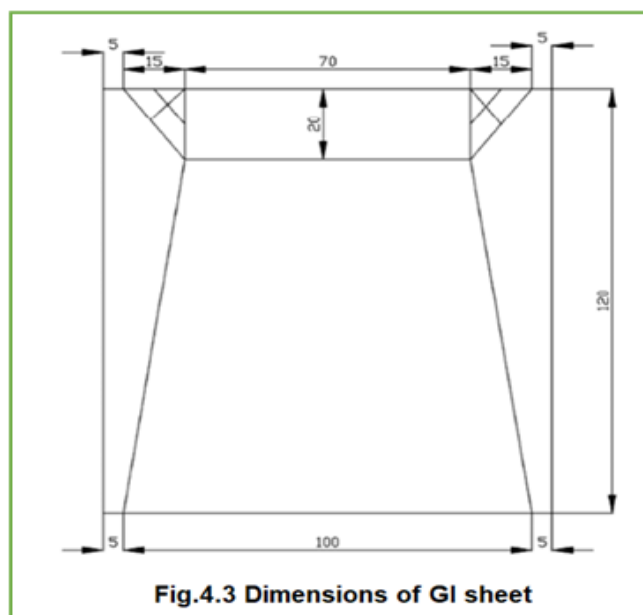


2. As illustrated in Fig.4.2, prepare the developing surface and create cylindrical tin.

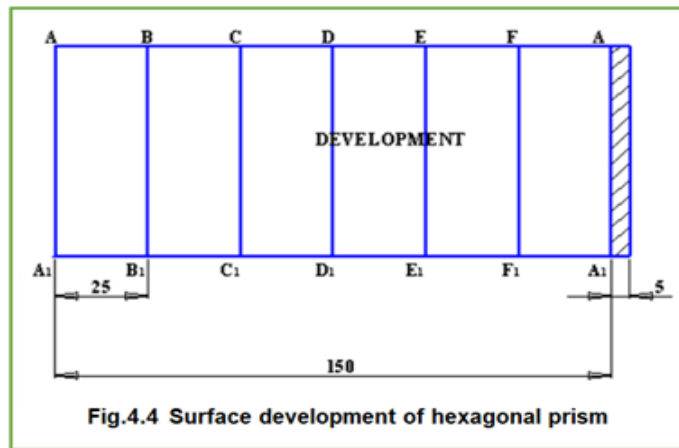


Try

1. Construct the open scoop as depicted in Fig. 4.3



2. create the hexagonal prism as shown in Fig.4.4

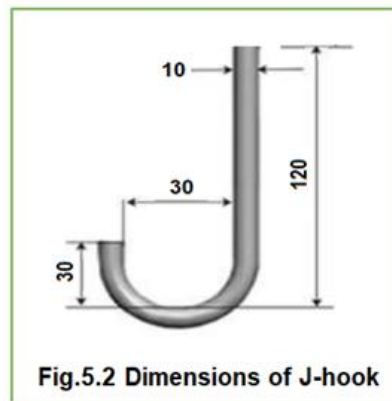
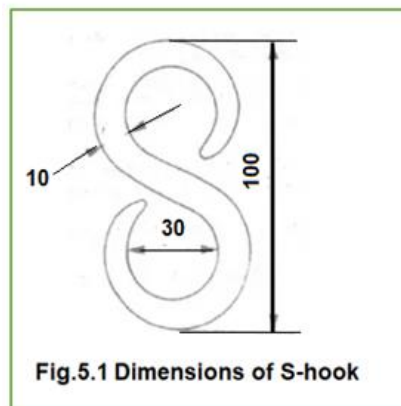


5. Introduction to black smithy

Black smithy or Forging is an oldest shaping process used for the producing small articles for which accuracy in size is not so important. The parts are shaped by heating them in an open fire or hearth by the blacksmith and shaping them through applying compressive forces using hammer. Thus forging is defined as the plastic deformation of metals at elevated temperatures into a predetermined size or shape using compressive forces exerted through some means of hand hammers, small power hammers, die, press or upsetting machine. It consists essentially of changing or altering the shape and section of metal by hammering at a temperature of about 980°C , at which the metal is entirely plastic and can be easily deformed or shaped under pressure. The shop in which the various forging operations are carried out is known as the smithy or smith's shop.

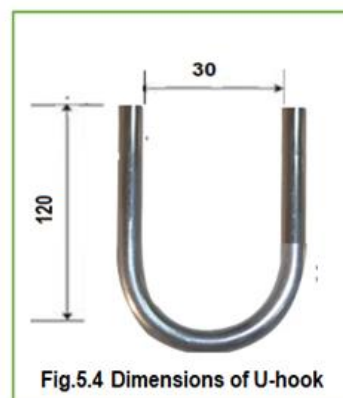
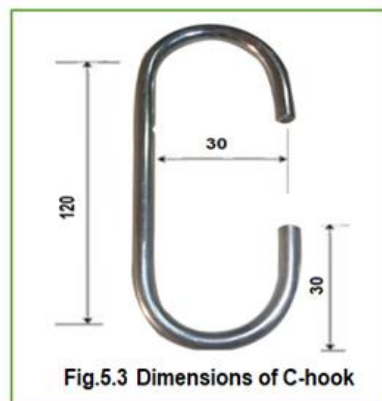
5.1. Experiments on black smithy

1. Make the s-hook as depicted in Fig. 5.1 using the mild steel rod provided.
2. Construct the J-hook using the given mild steel rod as indicated in Fig. 5.2.



Try

1. Create the C - hook with the given mild steel rod as shown in Fig. 5.3
2. Prepare the U - bend with the given mild steel rod as shown in Fig. 5.4

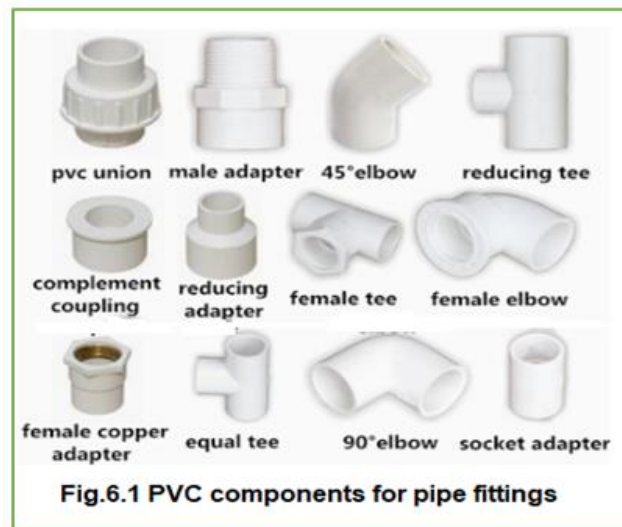


6. Introduction to plumbing

Plumbing is a skilled trade of working with pipes or tubes and plumbing fixtures. The process is mainly used for the supply of drinking water and the drainage of waste water, sometimes mixed with waste floating materials in a living or working place. A plumber is someone who installs or repairs piping systems, plumbing fixtures and equipment such as valves, washbasins, water heaters, waterclosets, etc. Thus it usually refers to a system of pipes and fixtures installed in a building for the distribution of water and the removal of waterborne wastes.

6.1. Experiments and demonstration on plumbing

1. Form of PVC pipe fitting through various components as shown in Fig. 6.1



2. Form of GI

shown in Fig. 6.2

pipe fitting with various components, as



Try

1. Form of PVC pipe fitting with reducer for water tap with different components as shown in Fig. 6.1
2. Form of GI pipe fitting with different components as shown in Fig. 6.2 for different fluids.

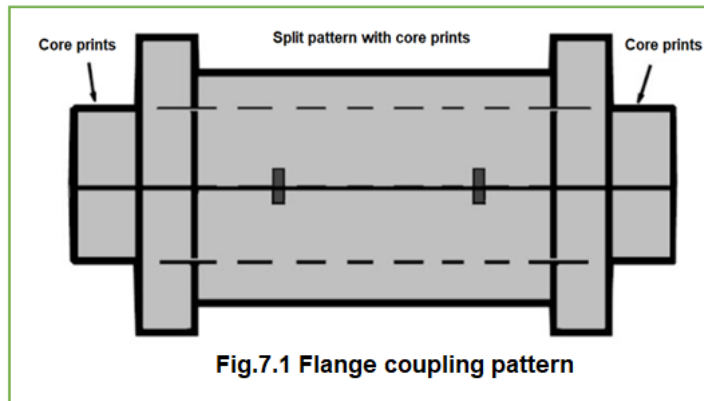
7. Introduction to moulding

Moulding is the process of manufacturing by shaping liquid or pliable raw material using a rigid frame called a mold or matrix. This itself may have been made using a pattern or model of the final object.

A mould is a hollowed-out block that is filled with a liquid or pliable material such as plastic, glass, metal, or ceramic raw material. The liquid hardens or sets inside the mould, adopting its shape. A mould is a counterpart to a cast. The very common bi-valve moulding process uses two moulds, one for each half of the object.

7.1. Experiments on mechanical components moulding (casting process)

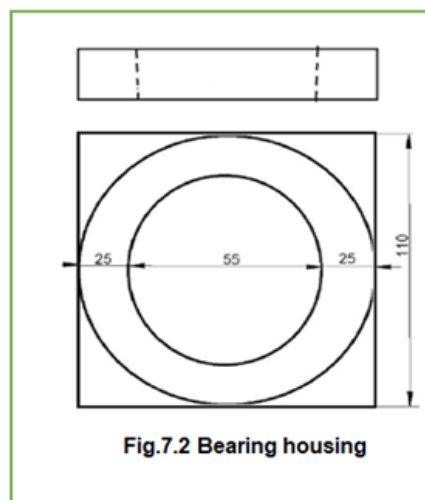
1. Making of flange mould using a given pattern as shown in Fig.7.1



2.

bearing housing mould as shown in Fig. 7.2.

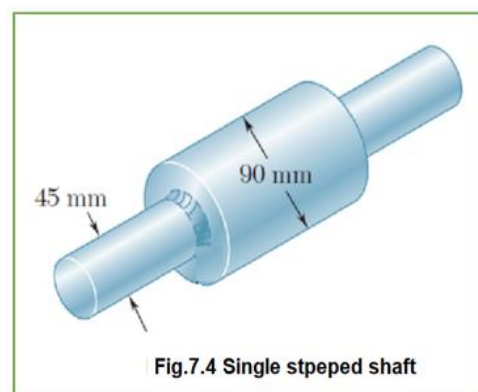
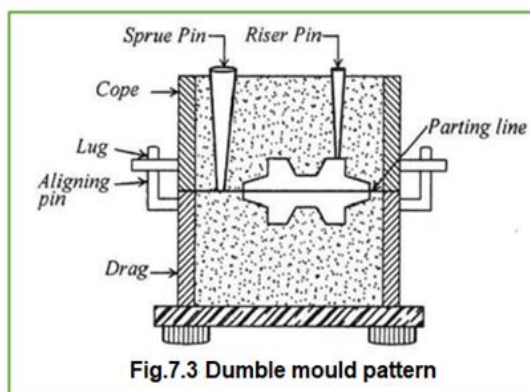
Utilizing the provided pattern, create the



Try

1. Making of dumbbell as shown in Fig.7.3
2. Using a single piece pattern, create a one-stepped shaft as shown in Fig. 7.4.

using a given pattern



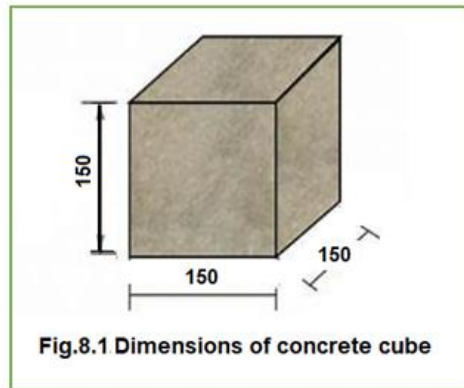
8. Introduction to concrete moulding and plaster of paris

Concrete is characterized by the type of aggregate or cement used, by the specific qualities it manifests, or by the methods used to produce it. In ordinary structural concrete, the character of the concrete is largely determined by a water-to-cement ratio. The lower the water content, all else being equal, the stronger the concrete. The mixture must have just enough water to ensure that each aggregate particle is completely surrounded by the cement paste, that the spaces between the aggregate are filled, and that the concrete is liquid enough to be poured and spread effectively.

Plaster of Paris is a white powder made from gypsum that mixes with water to form a paste that hardens quickly and is used chiefly for casts and moulds. It can be effectively worked with metal apparatuses or even abrasive sheets and can be shaped as per requirements. It is often applied in the form of a quick-setting paste with water.

8.1. Experiment on concrete/cement cube moulding and demonstration on plaster of paris mould making

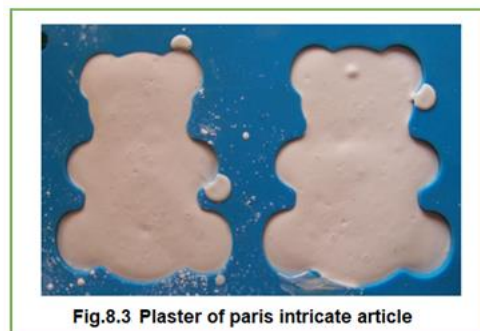
1. Preparation of concrete cube by moulding technique as shown in Fig.8.1



2. Demonstration on plaster of paris mould making.

Try

1. Preparation of any house hold specimens by plaster of paris mould making as shown in Fig. 8.2
2. Preparation of any intricate article by plaster of paris mould making as shown in Fig. 8.3



9. Introduction to hard soldering

Hard (silver) soldering ($>450^{\circ}\text{C}$) – Brass or silver is the bonding metal used in this process, and requires a blowtorch to achieve the temperatures at which the solder metals. Hard soldering is used to join precision components such as ferrous, brass, and copper.

9.1. Experiments on hard soldering

1. Soldering of two mild steel plates as shown in Fig. 9.1
2. Hard soldering of engine valve tappet as shown in Fig. 9.2



Fig.9.1 Soldering of mild steel plates



Fig.9.2 Engine valve tappet

Try

1. Hard soldering of copper with brass as shown in Fig.9.3
2. Hard soldering of stainless steel with brass as shown in Fig.9.4



Fig.9.3 Hard soldering of copper with brass



Fig.9.4 Hard soldering of stainless steel with brass

10. Demonstration on Computer Numerically Controlled (CNC) lathe

CNC turning is a highly precise and efficient subtractive machining process that works on the principle of the lathe machine. It involves placing the cutting tool against a turning workpiece to remove materials and give the desired shape.



Fig.10.1 CNC lathe

1. Demonstration of the plain turning process on a CNC lathe as shown in Fig.10.1
2. Demonstration of facing operations on a CNC lathe as shown in Fig.10.1.

11. Demonstration on Computer Numerically Controlled (CNC) milling

CNC milling involves cutting a prismatic workpiece using multipoint cutting tools producing precision components used in automotive and aeronautical industries.



Fig.11.1 CNC machining centre

1. Demonstration of plain milling (facing) on CNC milling as shown in Fig.11.1
2. Demonstration of precision slotting on CNC milling as shown in Fig.11.1.

12. Demonstration on 3D printing machine

3D printing or additive manufacturing enables to produce geometrically complex objects, shapes and textures. It often uses less material than traditional manufacturing methods and allows the production of prototypes / products that are not possible to produce economically with conventional manufacturing.

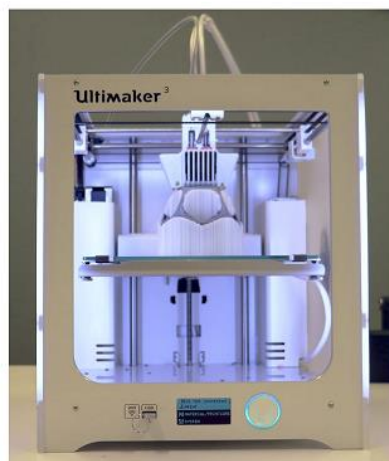


Fig.12.1 3D printer

1. Demonstration of 3D printing machine as shown in Fig.12.1 using Acrylonitrile butadiene styrene (ABS) material
2. Demonstration of 3D printing machine as shown in Fig.12.1 using Polylactic acid (PLA) material.

13. Demonstration on 6- axis robot

Robots have seen in recent years an expansion of their field of use with new requirements related to the increasing use of composites. The robots are then considered for machining operations (polishing, cutting, drilling etc.) that require high performance in terms of position, orientation, followed by trajectory precision and stiffness. The evolution of the performance of robots and programming software provides new machining

solutions. For complex parts, six axis robots offer more accessibility than a machining center CNC 5 axis and allow the integration of additional axes to extend the workspace.



Fig.13.1 6-Axis robot

1. Demonstration of the 6 – axis aristo robot as shown in Fig.13.1.
2. Demonstration of aristo sim software for robot movements and control.

14. Demonstration on cylindrical grinding machine

Most commonly, cylindrical grinding is used for grinding pieces with a central axis of rotation, like rods and cylinders. This process involves using a cylindrical grinder, which is a type of machinery categorized by rotation style and wheel device.

A grinding machine uses an abrasive product usually a rotating wheel to shape and finish a workpiece by removing metal and generating a surface within a given tolerance. A grinding wheel is made with abrasive grains bonded together. Each grain acts as a cutting tool, removing tiny chips from the workpiece.



Fig.14.1 Cylindrical grinding machine

1. Demonstration of grinding process on a cylindrical grinding machine as shown in Fig.14.1
2. Demonstration of shaft grinding process on a cylindrical grinding machine as shown in Fig.14.1

V. TEXT BOOKS:

1. Hajra Choudhury S.K., Hajra Choudhury A.K. and NirjharRoy S.K., “*Elements of Workshop Technology*”, Media promoters and publishers private limited, Mumbai, 4th Edition ,2020.
2. Kalpakjian S, Steven S. Schmid, “*Manufacturing Engineering and Technology*”, Pearson Education India Edition, 7th Edition, 2019.
3. Gowri P. Hariharan, A. Suresh Babu,” *Manufacturing Technology – I*”, Pearson Education, 3rd Edition,

2018.

VI. REFERENCE BOOKS:

1. Gowri P. Hariharan, A. Suresh Babu, “*Manufacturing Technology – I*”, Pearson Education, 5th Edition, 2018.
2. Roy A. Lindberg, “*Processes and Materials of Manufacture*”, Prentice Hall India, 4th Edition, 2017.
3. Rao P.N., “*Manufacturing Technology*”, Vol. I and Vol. II, Tata McGraw-Hill House, 2017.

VII. ELECTRONICS RESOURCES:

1. <https://elearn.nptel.ac.in/shop/iit-workshops/ongoing/additive-manufacturing-technologies-for-practicing-engineers/>.
2. https://akanksha.iare.ac.in/index?route=course/details&course_id=337

VIII. MATERIALS ONLINE:

1. Course Template
2. Laboratory manual

COURSE CONTENT

ESSENTIALS OF INNOVATION								
I Semester: AE / ME / CE / ECE / EEE / CSE (AI&ML) / IT								
II Semester: CSE / CSE (DS) / CSE (CS)								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
ACSD03	Foundation	L	T	P	C	CIA	SEE	Total
		-	2	-	1	40	60	100
Contact Classes: Nil	Tutorial Classes: Nil	Practical Classes: 45			Total Classes: 45			
Prerequisite: There are no prerequisites to take this course								

I. COURSE OVERVIEW:

Essentials of Innovation and Design thinking is a strategic approach towards creative problem-solving by placing users'/customers' needs above everything else. It is a process of questioning: questioning the problem, questioning assumptions, and questioning the implications. As a process it is a great catalyst of change and evolution. A Design thinking approach helps develop and build a culture of innovation across the students.

II. COURSE OBJECTIVES:

- I. The implications of disruption and the role of innovation.
- II. The various frameworks, tools, and techniques of design thinking.
- III. How to design, develop and implement an innovation product or service or process.

III. COURSE CONTENT:

Module-I; Philosophy of Innovation and Design Thinking

- Introduction to Innovation and Design Thinking
- History and Philosophy of Design Thinking
- Design Thinking as Problem-Solving Tool
- Design Thinking and its Benefits
- Design Thinking Mind-set

Module-2: Mechanics of Innovation and Design Thinking

- Integrative View of Design Thinking
- Design Thinking Process
- 5 Stages (Empathise, Define, Ideate, Prototype and Test)
- Conceptual Frameworks Used in Design Thinking Process
- Case Studies

Module-3: Design Thinking for Understanding Customers

- Understanding the User and Context
- Market Research
- Visualization and Customer Journey Mapping
- Empathy Mapping
- Redefining Problems, Brainstorming
- Reframing the Perspectives
- Ideation and Creativity
- Creative Ideation Methodologies
- Sketching & Visualization
- Storytelling

Module-4: Implementing Design Thinking

- Innovating Products, Services and Business Models
- Concept Evaluation and Concept Development
- Applications of Design Thinking
- Designing for Tangibles and Intangibles
- Ideas and Opportunities for Products

Module-5: Innovation Management

- Introduction to Innovation Management
- Business, Product & Process Innovation
- Organization Innovation
- Innovating Products, Services and Business Models
- Crafting a Better World Using Design Thinking & Innovation
- Design Thinking, Innovation and Organization Strategy
- Idea Pitching and Validation

Text Books:

- I. Nigel Cross, “*Design Thinking: Understanding How Designers Think and Work*”, Kindle Edition, 2011.
- II. Tim Brown, Harper Bollins, “*Change by Design*”, 2009.
- III. Idris Mootee, “*Design Thinking for Strategic Innovation*”, John Wiley & Sons, 2013.

COURSE CONTENT

ENVIRONMENTAL SCIENCE								
I Semester: AE / ME / CE / ECE / EEE/ CSE (AI & ML) / CSE / CSE (CS) / CSE (DS) / IT								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AHSD06	Foundation	L	T	P	C	CIA	SEE	Total
		-	-	-	-	-	-	-
Contact Classes: Nil	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: Nil			
Prerequisite: Basic Principles of earth science.								

I. COURSE OVERVIEW:

This course is an interdisciplinary study which examines the interaction between humans and the environment, with specific reference to the effects of modern technological advances. The students will be able to understand the sustainable development, ecological sustainability, environmental pollution, environmental issues in order to protect the environment and followed by the application of this knowledge to current environmental problems in the later years.

II. COURSES OBJECTIVES:

The students will try to learn

- The interrelationship between living organism and environment.
- The importance of environment by assessing its impact on the human world
- The knowledge on themes of biodiversity, natural resources, pollution control and waste management.
- The sustainability and unsustainability of various interactions between human society and the earth's natural systems

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO1 Infer the basic ecological principles, biogeochemical cycles and its function for the flow of energy in ecosystem.
- CO2 Awareness on different natural resources and its conservation for sustainable development.
- CO3 Use alternate energy resources for future growing energy needs.
- CO4 Predict the of importance of biodiversity for its productive use.
- CO5 Identify the global environmental problems by different types of environmental Pollution and international summits for minimizing the problems.
- CO6 Outline the features of laws and rules related to environment protection, environmental impact assessment towards sustainable development.

IV. SYLLABUS:

MODULE-I: ECOSYSTEMS

Environment: definition, scope and importance of ecosystem, classification, structure and function of an ecosystem, food chains, food webs and ecological pyramids, flow of energy; biogeochemical cycles, hydrological cycle, phosphorous cycle, nitrogen cycle, biomagnifications.

MODULE-II: NATURAL RESOURCES

Natural resources: classification of resources, living and nonliving resources; water resources: use and over utilization of surface and ground water, floods and droughts, dams, benefits and problems; mineral resources: use and exploitation, environmental effects of extracting and using mineral resources; land resources; energy resources: renewable and non-renewable energy sources, use of alternate energy source.

MODULE-III: BIODIVERSITY AND BIOTIC RESOURCES

Biodiversity and biotic resources: introduction, definition, genetic, species and ecosystem diversity; value of biodiversity: consumptive use, productive use, social, ethical, aesthetic and optional values; Hot spots of biodiversity.

Threats to biodiversity: habitat loss, poaching of wildlife, human-wildlife conflicts; Conservation of biodiversity: In situ and ex situ conservation.

MODULE-IV: ENVIRONMENTAL POLLUTION AND CONTROL TECHNOLOGIES

Environmental pollution: definition, causes, effects and control measures of air pollution, water pollution, soil pollution, impacts of modern agriculture and noise pollution; solid waste: municipal solid waste management, composition and characteristics of e-waste and its management; Pollution control technologies: waste water treatment methods, primary, secondary and tertiary; global environmental issues and global efforts: climate change and impacts on human environment, ozone depletion, ozone depleting substances; International conventions / protocols: Kyoto protocol and Montreal protocol.

MODULE-V: ENVIRONMENTAL POLICY AND LEGISLATION

Environmental legislations: environmental protection act, air act 1981, water act, forest act. municipal solid waste management and handling rules, biomedical waste management and handling rules, hazardous waste management and handling rules, population and its explosion.

V. TEXT BOOKS:

1. Erach Bharucha, *Text Book of Environmental Studies for Under Graduate Course*, Orient Black Swan, 3rd Edition, 2021.
2. Anubha Kaushik and C P Kaushik, *Perspectives in Environmental Studies*, New Age International private limited, New Delhi, 7th Edition, 2021.
3. Benny Joseph, *Environmental Studies*, Tata Mc Graw Hill Publishing Co. Ltd, New Delhi, 3rd Edition, 2017.

VI. REFERENCE BOOKS:

1. Dr.M Anji Reddy, *Text Book of Environmental Science and Technology*, BS Publications, 3rd Edition, 2014.
2. Y Anjaneyulu, *Introduction to Environmental Science*, BSP Books Private Limited, 3rd Edition, 2020.

VII. ELECTRONICS RESOURCES:

1. <https://www.meripustak.com/Environmental-Science-Isv-8th-Edition-121505>
2. https://www.meripustak.com&gclid=CjwKCAjwtp2bBhAGEiwAOZZTuFwLEkGc6SGNUZjXpz0ffeNwgBOHWQIKge-E-9UvXxTPxQJdjaTgJBoCrQIQAvD_BwE

VIII. MATERIALS ONLINE

1. Course Template
2. Tutorial Question Bank
3. Model Question Paper – I
4. Model Question Paper - II
5. Lecture Notes
6. Early Lecture Readiness Videos
7. Power Point Presentation



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

ENGINEERING CHEMISTRY								
I Semester: CSE / CSE (CS) / CSE (DS)								
II Semester: AE / ME / CE / ECE / EEE / CSE (AI & ML) / IT								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AHSD03	Foundation	L	T	P	C	CIA	SEE	Total
		3	-	-	3	40	60	100
Contact Classes: 64	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: 64			
Prerequisite: Basic principles of chemistry								

I. COURSE OVERVIEW:

This course focuses on the fundamental concepts of chemistry and then builds an interface with their industrial applications. The basic knowledge on chemical bonding and intermolecular forces which together are responsible for determining the properties of materials. The students will be able to analyze water purification processes to avoid industrial interruptions. The course concludes with an overview of involving electron transfer, including their applications in corrosion and energy storage for portable electronic devices. It should cultivate in students to identify chemistry in each piece of finely engineered products used in households and industry.

II. COURSES OBJECTIVES:

The students will try to learn

- The concepts of electrochemical principles and causes of corrosion in the new developments and breakthroughs efficiently in engineering and technology.
- The different parameters to remove causes of hardness of water and their reactions towards complexometric method.
- The properties, separation techniques of natural gas and crude oil along with potential applications in major chemical reactions.
- The different types of materials with respect to mechanisms and its significance in industrial applications.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO1 Acquire the basic knowledge of electrochemical principles related to corrosion and its control
- CO2 Interpret the basic properties of water for its usage in industrial and domestic applications.
- CO3 Use complexometry for calculation of hardness of water to avoid industrial problems.
- CO4 Extend the applications of polymers based on their degradability and properties.
- CO5 Choose the appropriate fuel based on their calorific value for energy efficient processes.
- CO6 Predict the knowledge on viability of advanced materials for technological improvements in various sectors.

IV. COURSE CONTENT:

MODULE-I: BATTERIES CHEMISTRY AND CORROSION (10)

Introduction to electrochemical cells: Galvanic cell, electrolytic cell; electrochemical series and its applications; Batteries: classification of batteries, construction, working and applications of Zinc-air battery, Lead-acid battery, Li-ion battery, applications of Li-ion battery to electric vehicles; Corrosion: causes and effects of corrosion, theories of chemical and electrochemical corrosion, mechanism of electrochemical corrosion; Corrosion control methods: cathodic protection, sacrificial anode and impressed current methods; Metallic coatings: Galvanization and tinning, electroplating of Copper.

MODULE-II: WATER AND ITS TREATMENT (09)

Introduction: Hardness of water, causes of hardness; types of hardness, temporary and permanent hardness, expression and units of hardness; estimation of hardness of water by complexometric method; potable water and its specifications, steps involved in the treatment of water, disinfection of water by chlorination and ozonation; external treatment of water; ion-exchange process; desalination of water: reverse osmosis, numerical problems.

MODULE-III: POLYMER TECHNOLOGY (10)

Polymers: classification of polymers; types of polymerization-addition, condensation polymerization with examples. Plastics: thermoplastic and thermosetting plastics; preparation, properties and engineering applications of PVC, Nylon 6,6 and Bakelite.

Biodegradable polymers: polylactic acid and polyvinyl alcohol and their applications. Elastomers: Introduction to natural rubber, vulcanization of natural rubber, preparation, properties and engineering applications of Buna-S and Thiokol rubber.

MODULE-IV: ENERGY SOURCES (09)

Introduction to fuels; classification of fuels; Solid fuels: coal; analysis of coal, proximate and ultimate analysis and their significance; Liquid fuels: petroleum and its refining; Gaseous fuels: composition, characteristics and applications of natural gas, LPG and CNG; Alternative and non-conventional sources of energy: solar, wind and hydropower advantages and disadvantages. Calorific value of fuel: HCV and LCV, Dulong's formula, calculation of air quantity required for complete combustion of fuel, numerical problems.

MODULE-V: ENGINEERING MATERIALS (10)

Nanomaterials: Introduction, preparation of nanoparticles by sol-gel method, chemical reduction method

and applications of nanomaterials. Smart materials and their engineering applications: shape memory

materials, Poly L-Lactic acid. Thermoresponsive materials: Polyacryl amides, Poly vinyl amides.

Cement: composition of Portland cement, setting and hardening of cement.

Lubricants: characteristics of a good lubricant, mechanism of lubrication, thick film, thin film and extreme

pressure lubrication; properties of lubricants: viscosity, flash and fire point, cloud and pour point.

V. TEXT BOOKS:

1. JAIN & JAIN, P.C. Jain, Monika Jain, *Engineering Chemistry*, Dhanpat Rai publishing Company (P) limited, 17th edition, 2022.

VI. REFERENCE BOOKS:

1. Shashi Chawla, *Text Book of Engineering Chemistry*, Dhanat Rai and Company (P) Limited, 1st Edition, 2017.
2. Jayashree Anireddy, *Textbook of Engineering chemistry*, Wiley Publications, 2023
3. S.S.Dara, *Text of Engineering Chemistry*, S.Chand & Co, New Delhi, 12th edition, 2018.
4. Nitin K Puri, *Nanomaterials Synthesis Properties and Applications*, I K international publishing house pvt Ltd, 1st edition 2021.

VII. ELECTRONICS RESOURCES:

1. Engineering chemistry (NPTEL Web-book), by B. L. Tembe, Kamaluddin and M. S.Krishnan.http://www.cdeep.iitb.ac.in/webpage_data/nptel/Core%20Science/Engineering%20Chemistry%201/About-Faculty.html
2. https://books.google.co.in/books?id=R1JtyILNIsAC&pg=PR3&source=gbs_selected_page&cad=3#v=onepage&q&f=false
3. https://books.google.co.in/books?id=eQTLcGAAQBAJ&pg=SA1PA53&source=gbs_selected_pages&cad=3#v=onepage&q&f=false

VIII. MATERIALS ONLINE

1. Course Template
2. Tutorial Question Bank
3. Tech talk Topics
4. Assignments
5. Definition and Terminology
6. Model Question Paper – I
7. Model Question Paper - II
8. Lecture Notes
9. Early Lecture Readiness Videos
10. Power point presentation

APPLIED PHYSICS								
I Semester: CSE / CSE (CS) / CSE (DS)								
II Semester: AE / ME / CE / ECE / EEE / CSE (AI & ML) / IT								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AHSD07	Foundation	L	T	P	C	CIA	SEE	Total
		3	-	-	3	40	60	100
Contact Classes: 48	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: 48			
Prerequisite: Basic principles of physics								

I. COURSE OVERVIEW:

The aim of this course is to enhance understanding of fundamental knowledge in physics needed for the future technological advances. The framework prepares students to engage in scientific questioning and extend thinking to investigations. The concepts cover current topics in the fields of solid state physics, modern physics, superconductors and nanotechnology. This knowledge helps to develop the ability to apply the principles in many technological sectors such as nanotechnology, optical fiber communication, quantum technology etc.

II. COURSES OBJECTIVES:

The students will try to learn

- V. Fundamental concepts needed to explain a crystal structure in terms of atom positions, unit cells, and crystal symmetry.
- VI. Basic formulations in wave mechanics for the evolution of energy levels and quantization of energies for a particle in a potential box with the help of mathematical description.
- VII. The metrics of optoelectronic components, lasers, optical fiber communication and be able to incorporate them into systems for optimal performance.
- VIII. The appropriate magnetic, superconducting and nanomaterials required for various engineering applications.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO 1 Use the general rules of indexing of directions and planes in lattices to identify the crystal systems and the Bravais lattices.
- CO 2 Use the concepts of dual nature of matter and Schrodinger wave equation to a particle enclosed in simple systems.
- CO 3 Analyze the concepts of laser with normal light in terms of mechanism for applications in different fields and scientific practices.
- CO 4 Strengthen the knowledge on functionality of components in optical fiber communication system by using the basics of signal propagation, attenuation and dispersion.
- CO 5 Gain deeper understanding on properties of magnetic and superconducting materials suitable for engineering applications.
- CO 6 Review the principle factors, fabrication, characterization techniques and the applications of nanomaterials.

IV. COURSE CONTENT:

MODULE - I: CRYSTAL STRUCTURES (10)

Introduction, space lattice, basis, unit cell, lattice parameter, Bravais lattices, crystal systems, structure and packing fractions of simple cubic, body centered cubic, face centered cubic crystals, directions and planes in crystals, Miller indices, separation between successive [h k l] planes.

MODULE –II: QUANTUM PHYSICS (09)

Waves and particles, de Broglie hypothesis, matter waves, Davisson and Germer's experiment, Schrödinger's time independent wave equation, physical significance of the wave function, infinite square well potential.

MODULE –III: LASERS AND FIBER OPTICS (10)

Characteristics of lasers, spontaneous and stimulated emission of radiation, population inversion, lasing action, Ruby laser, He-Ne laser, applications of lasers.

Principle and construction of an optical fiber, acceptance angle, numerical aperture, types of optical fibers

(Single mode, multimode, step index, graded index), optical fiber communication system with block diagram, applications of optical fibers.

MODULE –IV: MAGNETIC AND SUPERCONDUCTING PROPERTIES (10)

Permeability, field intensity, magnetic field induction, magnetization, magnetic susceptibility, origin of magnetic moment, Bohr magneton, classification of dia, para and ferro magnetic materials on the basis of magnetic moment, Hysteresis curve.

Superconductivity, general properties, Meissner effect, effect of magnetic field, type-I & type-II superconductors, BCS theory, applications of superconductors.

MODULE –V: NANOTECHNOLOGY (09)

Nanoscale, quantum confinement, surface to volume ratio, bottom-up fabrication: Sol-gel, precipitation, combustion methods, top-down fabrication: Ball milling, physical vapor deposition, chemical vapor deposition, characterization techniques: X-ray diffraction, transmission electron microscopy, applications of nanomaterials.

V. TEXT BOOKS:

2. Arthur Beiser, Shobhit Mahajan and Rai Choudhary, *Concepts of Modern Physics*, Tata McGraw Hill, 7th Edition, 2017.

VI. REFERENCE BOOKS:

2. H J Callister, *A Textbook of Materials Science and Engineering*, Wiley Eastern Edition, 8th Edition, 2013.
3. Halliday, Resnick and Walker, *Fundamentals of Physics*, John Wiley & Sons, 11th Edition, 2018.
4. Charles Kittel, *Introduction to Solid State Physics*, Wiley Eastern, 2019.
5. S.L. Gupta and V. Kumar, *Elementary Solid State Physics*, Pragathi Prakashan, 2019.
6. K K Chattopadhyay and A N Banerjee, *Introduction to Nanoscience and Nanotechnology*, Prentice Hall India, 2nd Edition, 2011.

VII. ELECTRONICS RESOURCES:

4. NPTEL :: Physics - NOC: Quantum Mechanics I
5. NPTEL :: Physics - NOC: Introduction to Solid State Physics
6. NPTEL :: Physics - NOC: Solid State Physics
7. <https://nptel.ac.in/courses/104104085>
8. NPTEL :: Metallurgy and Material Science - NOC: Nanotechnology, Science and Applications

VIII. MATERIALS ONLINE

11. Course template
12. Tutorial question bank
13. Definition and terminology
14. Tech-talk topics
15. Assignments
16. Model question paper - I
17. Model question paper - II
18. Lecture notes
19. Early learning readiness videos (ELRV)
20. Power point presentations

COURSE CONTENT

DIFFERENTIAL EQUATIONS AND VECTOR CALCULUS								
II Semester: AE / ME / CE / ECE / EEE / CSE / CSE (AI & ML) / CSE (DS) / CSE (CS) / IT								
Course Code	Category	Hours/Week			Credits	Maximum Marks		
AHSD08	Foundation	L	T	P	C	CIA	SEE	Total
		3	1	-	4	40	60	100
Contact Classes: 48	Tutorial Classes: 16	Practical Classes: Nil			Total Classes: 64			
Prerequisite: Basic Principles of Matrices and Calculus								

I. COURSE OVERVIEW:

This course serves as a foundation course on differential equations and vector calculus. It includes techniques for solving ordinary differential equations, partial differential equations, vector differentiation and vector integration. It is designed to extract the mathematical developments, skills, from basic concepts to advance level of engineering problems to meet the technological challenges.

II. COURSE OBJECTIVES:

The students will try to learn:

- I The analytical methods for solving first and higher order differential equations with constant coefficients.
- II The analytical methods for formation and solving partial differential equations.
- III The physical quantities of vector valued functions involved in engineering field.
- IV The logic of vector theorems for finding line, surface and volume integrals.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO1 Utilize the methods of differential equations for solving the orthogonal trajectories and Newton's law of cooling.
- CO2 Solve the higher order linear differential equations with constant coefficients by using method of variation of parameters.
- CO3 Make use of analytical methods for PDE formation to solve boundary value problems.
- CO4 Identify various techniques of Lagrange's method for solving linear partial differential equations which occur in science and engineering.
- CO5 Interpret the vector differential operators and their relationships for solving engineering problems.
- CO6 Apply the integral transformations to surface, volume and line of different geometrical models in the domain of engineering.

IV. COURSE CONTENT:

MODULE-I: FIRST ORDER AND FIRST DEGREE ODE (10)

Exact differential equations, Equations reducible to exact differential equations, linear and Bernoulli's equations, Applications: Orthogonal Trajectories (Cartesian Coordinates) Newton's law of cooling.

MODULE-II: ORDINARY DIFFERENTIAL EQUATIONS OF HIGHER ORDER (10)

Second order linear differential equations with constant coefficients: non-homogeneous terms of the type e^{ax} , $\sin ax$, $\cos ax$, polynomials in x , $e^{ax}(x)$ and method of variation of parameters.

MODULE-III: PARTIAL DIFFERENTIAL EQUATIONS (09)

Formation of partial differential equations by elimination of arbitrary constants and arbitrary functions.

Solutions of first order linear equations: method of grouping and method of multipliers.

MODULE-IV: VECTOR DIFFERENTIATION (09)

Scalar and vector point functions; definitions of gradient, divergent and curl with examples; solenoidal and irrotational vector point functions; scalar potential function.

MODULE–V: VECTOR INTEGRATION (10)

Line integral, surface integral and volume integral, Green's theorem in a plane, Stoke's theorem and Gauss divergence theorem without proofs.

V. TEXT BOOKS:

1. B. S. Grewal, *Higher Engineering Mathematics*, 44/e, Khanna Publishers, 2017.
2. Erwin Kreyszig, *Advanced Engineering Mathematics*, 10/e, John Wiley & Sons, 2011.

VI. REFERENCE BOOKS:

1. R. K. Jain and S. R. K. Iyengar, *Advanced Engineering Mathematics*, 3/ed, Narosa Publications, 5th Edition, 2016.
2. George B. Thomas, Maurice D. Weir and Joel Hass, Thomas, *Calculus*, 13/e, Pearson Publishers, 2013.
3. N.P.Bali and Manish Goyal, *A text book of Engineering Mathematics*, Laxmi Publications, Reprint, 2008
4. Dean G. Duffy, *Advanced Engineering Mathematics with MATLAB*, CRC Press.
5. Peter O'Neil *Advanced Engineering Mathematics*, Cengage Learning.
6. B.V. Ramana, *Higher Engineering Mathematics*, McGraw Hill Education.

VII. ELECTRONIC RESOURCES:

1. Engineering Mathematics - I, By Prof. Jitendra Kumar
[\[https://onlinecourses.nptel.ac.in/noc23_ma88/preview\]](https://onlinecourses.nptel.ac.in/noc23_ma88/preview)
2. Advanced Calculus for Engineers, By Prof. Jitendra Kumar, Prof. Somesh Kumar
https://onlinecourses.nptel.ac.in/noc23_ma86/preview
3. http://www.efunda.com/math/math_home/math.cfm
4. <http://www.ocw.mit.edu/resources/#Mathematics>
5. <http://www.sosmath.com>
6. <http://www.mathworld.wolfram.com>

VIII. MATERIAL ONLINE:

1. Course template
2. Tech-talk topics
3. Assignments
4. Definition and terminology
5. Tutorial question bank
6. Model question paper – I
7. Model question paper – II
8. Lecture notes
9. Early lecture readiness videos (ELRV)
10. Power point presentations

COURSE CONTENT

ENGINEERING MECHANICS								
II Semester: AE / ME / CE								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AMED04	Foundation	L	T	P	C	CIA	SEE	Total
		3	0	0	3	40	60	100
Contact Classes: 48	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes:48			
Prerequisite: Matrices and Calculus								

I. COURSE OVERVIEW:

Engineering Mechanics is a foundational course that introduces students to the principles of mechanics and their applications in analyzing and solving real-world engineering problems. This course focuses on imparting a comprehensive understanding of statics and dynamics, which are essential concepts for engineers across various disciplines. Through a combination of theoretical concepts, mathematical derivations, and practical applications, students will develop the skills necessary to analyze and predict the behavior of structures and systems under different conditions.

II. COURSES OBJECTIVES:

The students will try to learn

- I. The application of mathematics and science principles to represent the free body diagrams in the area of rigid body mechanics.
- II. The conditions of static and dynamic equilibrium of bodies subjected to a particular force system for solving the field problems.
- III. The effects of force and motion while carrying out the innovative design functions of engineering.

III. COURSE OUTCOMES:

After successful completion of the course, students should be able to:

- | | |
|-----|---|
| CO1 | Determine the unknown forces by free body diagrams to a given equilibrium force system through laws of mechanics. |
| CO2 | Calculate the system of forces acting on wedge and screw jack by using the laws of static and dynamic frictions. |
| CO3 | Use the concepts of centroid in stability problems for evaluation of area moment of inertia. |
| CO4 | Identify the mass moment of inertia of symmetrical and non-symmetrical section using the concepts of centre of gravity. |
| CO5 | Solve the position, velocity, acceleration and the characteristics of a body in dynamic equilibrium for various types of motion using appropriate mathematical tools. |
| CO6 | Develop the governing equation from first principles by using work - energy and impulse - momentum in dynamic equilibrium condition. |

IV. SYLLABUS:

MODULE – I: Introduction to Engineering Mechanics (10)

2D Force Systems: Basic concepts, particle equilibrium; rigid body equilibrium; system of forces, coplanar concurrent forces, resultant, moment of forces and its application; couples and resultant of force system, equilibrium of system of forces, free body diagrams, equations of equilibrium of coplanar systems.

MODULE – II: Friction, Centroid and Centre of gravity (8)

Friction: Types of friction, limiting friction, laws of friction, static and dynamic friction; motion of bodies, wedge friction, screw jack.

Centroid and Centre of Gravity: Centroid of lines, areas and volumes from first principle, centroid of composite sections; centre of gravity and its implications, theorems of Pappus–Guldinus.

MODULE – III: Area moment of inertia and Mass moment of inertia (8)

Area moment of inertia: Definition, moment of inertia of plane sections from first principles, theorems of moment of inertia, moment of inertia of standard sections and composite sections; product of inertia, parallel axis theorem, perpendicular axis theorem.

Mass Moment of Inertia: Moment of inertia of masses, transfer formula for mass moment of inertia, mass moment of inertia of composite bodies.

MODULE – IV: Kinematics of rigid bodies and Impulse – momentum method (10)

Review of particle dynamics, rectilinear motion; Plane curvilinear motion (rectangular path, and polar coordinates). Relative and constrained motion. Impulse-momentum (linear, angular); impact (direct and oblique).

MODULE – V: Kinetics of rigid bodies and Work – energy principle (12)

Kinetics of rigid bodies, basic terms, D’ Alembert’s principle and its applications in plane motion and connected bodies; instantaneous centre of rotation in plane motion and simple problems; work-kinetic energy, power, potential energy. work energy principle and its application in plane motion of connected bodies.

V. TEXT BOOKS:

4. K. Vijay Reddy, J. Suresh Kumar, “*Singer’s Engineering Mechanics Statics and Dynamics*”, B S Publishers, 1st edition, 2011.
5. S. Bhavikatti, “*A Text Book of Engineering Mechanics*”, New Age International, 5th edition, 2020.

VI. REFERENCE BOOKS:

4. S. Timoshenko, D. H. Young & J. V. Rao, “*Engineering Mechanics*”, 5th edition, TMH, 2017.
5. A. K. Tayal, “*Engineering Mechanics*”, Uma Publications, 14th edition, 2013.
6. R. K. Bansal “*Engineering Mechanics*”, Laxmi Publication, 8th edition, 2013.
7. Irving H. Shames, “*Engineering Mechanics*”, Prentice Hall, 4th edition, 2021.
8. R. C. Hibbler, “*Engineering Mechanics: Principles of Statics and Dynamics*”, Pearson Press, 5th edition, 2021.
9. Irving H. Shames (2006), “*Engineering Mechanics*”, Prentice Hall, 4th edition, 2013.

VII. ELECTRONICS RESOURCES:

1. <https://nptel.ac.in/courses/112106286>
2. https://akanksha.iare.ac.in/index?route=course/details&course_id=33
3. https://akanksha.iare.ac.in/index?route=course/details&course_id=31
4. https://akanksha.iare.ac.in/index?route=course/details&course_id=1293

VIII. MATERIALS ONLINE:

1. Course template
2. Tutorial question bank
3. Tech talk topics
4. Open end experiments
5. Definitions and terminology
6. Assignments
7. Model question paper - I
8. Model question paper - II
9. Lecture notes
10. E-learning readiness videos (ELRV)
11. Power point presentation

COURSE CONTENT

ENGINEERING CHEMISTRY LABORATORY								
I Semester: CSE / CSE (CS) / CSE (DS)								
II Semester: AE / ME / CE / ECE / EEE / CSE (AI&ML) / IT								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AHSD05	Foundation	L	T	P	C	CIA	SEE	Total
		-	-	3	1	40	60	100
Contact Classes: Nil	Tutorial Classes: Nil	Practical Classes: 36			Total Classes: 36			
Prerequisite: Basic Principles of Chemistry								

I. COURSE OVERVIEW:

The course encourages introducing analytical tools in an Engineering perspective. The course efforts to provide the basic knowledge of analytical methodology, outlines the importance of volumetric analysis, comprehensive instrumental analysis for properties of fuels, colorimetric analysis and spectroscopic analysis. This practical approach gives the essence of analytical chemistry for skill development in determinations of materials properties and its viability in the industry.

II. COURSES OBJECTIVES:

The students will try to learn:

- I. The quantitative analysis to know the strength of unknown solutions by instrumental methods.
- I. The troubles of hard water and its estimation by analytical techniques.
- II. The applications of appropriate lubricant for finely tuned machinery.
- III. The basic knowledge on quantity of light absorbed by the materials.

III. COURSE OUTCOMES:

After successful completion of the course students should be able to:

- CO1 Use conductivity meter and potentiometer for measurement of conductance and electromotive force of solutions
- CO2 Examine the corrosion tendency in metals and its control by using inhibitor.
- CO3 Make use of the principles of water analysis for domestic and industrial applications.
- CO4 Demonstrate the characteristics of different fuels for finding the calorific value.
- CO5 Use different types of lubricants to know its properties for the proper lubrication of machinery in industries.
- CO6 Interpret the absorption tendency of solids or liquids by using colorimetry and spectroscopy techniques.

IV. COURSE CONTENT:

1. GETTING STARTED EXERCISES

1.1 Introduction to Chemistry Laboratory

The fundamental concepts and theories required for carrying out qualitative and quantitative analysis.

Detailed explanation on the analytical techniques used for qualitative analysis. Emphasis on instrumental method of analysis and its advantages over conventional methods.

- i. Types of analysis
 - ii. Difference between qualitative and quantitative analysis
 - iii. Common techniques of qualitative and quantitative analysis
 - iv. Introduction to instrumental method of analysis
 - v. Introduction to basic techniques and handling of common apparatus
 - vi. Discussion of Material Safety Data Sheet (MSDS) of chemicals
 - vii. Identification of toxic signs and safety procedures of chemical laboratory
-

1.2 Safety Guidelines to Chemistry Laboratory

The chemistry laboratory must be a safe place in which to work and learn about chemistry.

- i. Wear a chemical-resistant apron.
 - ii. Be familiar with your lab work sheet before you come to lab. Follow all written and verbal instructions carefully. Observe the safety alerts in the laboratory directions. If you do not understand a direction or part of a procedure, ask the teacher before proceeding.
 - iii. When entering the laboratory room, do not touch any equipment, chemicals, or other materials without being instructed to do so. Perform only those experiments authorized by the instructor.
 - iv. If you take more of a chemical substance from a container than you need, you should not return the excess to the container. This might cause contamination of the substance remaining. Dispose of the excess as your instructor directs.
 - v. Never smell anything in the laboratory unless your teacher tells you it is safe. Do not smell a substance by putting your nose directly over the container and inhaling. Instead, waft the vapors toward your nose by gently fanning the vapors toward yourself.
 - vi. Do not directly touch any chemical with your hands. Never taste materials in the laboratory.
 - vii. Work areas should be kept clean and tidy at all times. Always replace lids or caps on bottles and jars.
-

1.3 Data recording and reports

Students must record their experimental values in the provided tables in this laboratory manual and reproduce them in the laboratory worksheets. Worksheets are integral to recording the methodology and results of an experiment. In engineering practice, the laboratory worksheets serve as a valuable reference to the technique used in the laboratory. Note that the data collected will be an accurate and permanent record of the data obtained during the experiment and the analysis of the results.

2. CONDUCTOMETRY

2.1 Determine the neutralization point between strong acid against strong base

- i. **The basic principle of conductometric titrations**
 - ii. **Titration of unknown solution of acid with base**
 - iii. **Graphical plots on volume of titrant vs. conductance**
-

3. POTENTIOMETRY

3.1 Estimate the amount of Iron by using potentiometry

- i. **The basic principle of potentiometric titrations**
 - ii. **Titration of Mohr's salt with potassium dichromate**
 - iii. **Graphical plots on volume of titrant vs. potential**
-

4. pH METRY

4.1 Determine the pH of the unknown solution by pH metry

- i. **The basic principle of pH metry**
 - ii. **Titration of unknown solution with standard acid**
 - iii. **Graphical plots on volume of titrant vs. pH to obtain equivalence point**
-

5. ARGENTOMETRIC TITRATIONS

5.1 Determination of chloride content of water by argentometry

- i. **Principle of Argentometric titration.**
 - ii. **Titration of water samples by using EDTA to find the total hardness in water.**
-

6. MEASUREMENT OF TOTAL DISSOLVED SOLIDS IN WATER

6.1 Measurement of total dissolved solids (TDS) in different water samples

- i. **Specifications of potable water**
 - ii. **Measure the total dissolved solids in different water samples by TDS meter**
-

7. COMPLEXOMETRY METHOD

7.1 Estimate the total hardness of water by EDTA

- i. **Principle of complexometric titration**
 - ii. **Titration of water samples by using EDTA to find the total hardness in water.**
-

8. BOMB CALORIMETER

8.1 Determine the calorific value of solid or liquid fuels using Bomb calorimeter

- i. Significance of bomb calorimeter
- ii. Combustion of solid or liquid fuels in bomb calorimeter to find its calorific value

9. VISCOSITY OF LUBRICANT

9.1 Determine the viscosity of the lubricants using Red Wood viscometer / Ostwald's viscometer

- i. The principle of viscosity of lubricant
- ii. Significance of viscosity index of lubricant
- iii. Viscosity of given lubricant at various temperature by using Red wood viscometer

10. FLASH AND FIRE POINTS OF LUBRICANT

10.1 Determine the flash and fire points of lubricants

- i. Significance of flash and fire point of lubricant in industries
- ii. Flash and Fire points of a given lubricant by using Pensky Martens flash point apparatus

11. CLOUD AND POUR POINTS OF LUBRICANT

11.1 Determine cloud and pour points of lubricants

- i. Significance of cloud and pour point of lubricants in industries
- ii. Cloud and Pour points of a given lubricant by using cloud and pour point apparatus

12. COLORIMETRY

12.1 Estimate the metal ion concentration using colorimeter

- i. Complexation of metal ion with ligands
- ii. Detection of absorbance of the colored metal -ligand complex solution
- iii. Graphical determination of concentration of the metal ions in the solution

13. SPECTROSCOPY

13.1 Characterization of nanomaterials by UV-visible spectrophotometer

- i. Synthesis of silver oxide nanomaterials
- ii. Dispersion of nanoparticles in suitable solvent
- iii. Determination of absorption edge of the nanoparticles using spectroscopic technique

V. TEXTBOOKS:

1. K. Mulkanti et al. *Practical Engineering Chemistry*, B.S. Publications, Hyderabad.
2. Vogel's, *Quantitative chemical analysis*, prentice Hall, 6th Edition, 2009.

VI. REFERENCE BOOKS:

1. Solanki, M. K. *Engineering Chemistry Laboratory Manual*. (Edu creation Publishing, 2019).
2. Jeffery, G. H. in *TEXTBOOK OF QUANTITATIVE CHEMICAL ANALYSIS* (ed John Wiley and Sons) (1989).
3. Gary-D-Christian, P. K. S. D., Kevin A. Schug. *Analytical-Chemistry-by-Gary-D-Christian*. 7 edn, Vol. 7 826 (Wiley, 2014).
4. Budinski, Kenneth G., *Engineering materials: properties and selection*, 5th edition, Prentice-Hall, 1996, pg.423.
5. *Engineering chemistry by Jain & Jain*, 17th Edition, ISBN:978-93-5216-641-1
6. Nitin K Puri, "Nanomaterials synthesis properties and applications" I K international publishing house Pvt Ltd, 1st Edition 2021.

7. B. Ramadevi and P. Aparna, S Chand publications, *lab manual for engineering chemistry*, S Chand publications, NewDelhi, 1st Edition 2022.

VII. ELECTRONICS RESOURCES:

1. <https://nptel.ac.in/translation>
2. <https://nptel.ac.in/courses/115105120>
3. <https://archive.nptel.ac.in/courses/122/101/122101001/#>

VIII. MATERIALS ONLINE

1. Course Template
2. Laboratory Manual

COURSE CONTENT

APPLIED PHYSICS LABORATORY								
I Semester: CSE / CSE (CS) / CSE (DS)								
II Semester: AE / ME / CE / ECE / EEE / CSE (AI & ML) / IT								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AHSD09	Foundation	L	T	P	C	CIA	SEE	Total
		-	-	2	1	40	60	100
Contact Classes: Nil	Tutorial Classes: Nil	Practical Classes: 45			Total Classes: 45			
Prerequisite: Basic Principles of Physics								

I. COURSE OVERVIEW:

The aim of the course is to provide hands on experience for experiments in different areas of physics. Students will be able to perform the experiments with interest and an attitude of learning. This laboratory includes experiments involving electromagnetism and optoelectronics. These also develop student's expertise in applying physical concepts to practical problem and apply it for different applications.

II. COURSES OBJECTIVES:

The students will try to learn:

- I Familiarize with the lab facilities, equipment, standard operating procedures.
- II The different kinds of functional magnetic materials which paves a way for them to use in various technical and engineering applications.
- III The analytical techniques and graphical analysis to study the experimental data for optoelectronic devices.
- IV The application of characteristics of lasers and its propagation in optical fiber communication.

III. COURSE OUTCOMES:

After successful completion of the course, students should be able to:

- CO1 Identify the type of semiconductor using the principle of Hall effect and also determine the energy gap and resistivity of a semiconductor diode using four probe methods.
- CO2 Illustrate principle, working and application of wave propagation and compare the results of frequency with theoretical harmonics and overtones.
- CO3 Investigate the energy losses, curie temperature and properties associated with a given Ferro magnetic material.
- CO4 Examine launching of light through optical fiber from the concept of light gathering capacity of numerical aperture and determine the divergence of Laser beam
- CO5 Graph V-I /L-I characteristics of various optoelectronic devices like Light Emitting diode, Solar cell at different intensities to understand their basic principle of functioning as well as to infer the value of Planck's constant.
- CO6 Analyze the variation of magnetic field induction produced at various points along the axis of current carrying coil.

IV. COURSE CONTENT:

1. GETTING STARTED EXERCISES

1.1 On Errors and Uncertainty in a measurement:

When a number represents a physical measurement, it is never exact because of the limitations of the instrument used or the way it was employed etc. It is essential, therefore, that each experimental result be presented in a way that indicates its reliability. The accuracy of result is important, for example, the calibration of the measuring instruments or systematic errors on the part of whoever is taking the data.

The following table is useful in thinking about these concepts:

Problem	Remedy
Mistakes and blunders	Repeat measurements several times to check yourself
Systematic errors	Use calibrated instruments properly and carefully
Random errors	Treat data statistically and report on the average magnitude of errors

1.2 Making a good graph:

- Keep your axes straight: If you need to plot "A vs B", or "A as a function of B", then A is on the vertical axis and B is on the horizontal axis.
 - The crucial part is choosing the range and scale for each axis. The range must be just large enough to accommodate all data and small enough that the scale is readable.
 - The scale should be spread out enough so that data take up most of the graph area and labeled so that plotting (and reading) is easy. Where appropriate, error bars should be included to indicate the uncertainty in measurements.
 - When a line is drawn, it should be a smooth one that best fits data. In general, there should be as many points on one side of the line as on the other. If data is taken properly, the line should pass inside of the error bars for each point.
 - If graph shows that one quantity is proportional to another, it should be a straight line that starts at the origin and passes through the plotted data with as many points on one side as the other.
 - If the slope of the line is to be found, choose two points on the line that are as far apart as possible. This will minimize the error that is introduced in reading the value of those points.
 - The slope is the difference between the vertical values of those points divided by the difference in the horizontal values of those points.
-

1.3 Data recording and worksheets

- Write the work sheets for the allotted experiment and keep them ready before the beginning of each lab.
- Perform the experiment and record the observations in the worksheets.
- Analyze the results and get the work sheets evaluated by the faculty.
- Upload the evaluated reports online from CMS LOGIN within the stipulated time.

2. HALL EFFECT (LORENTZ FORCE)

- 2.1 Study the phenomenon of Hall effect and determine the charge carrier density and Hall coefficient of a given sample.
- 2.2 Hint whether the given semiconductor is p - type or n - type using the principle of Hall Effect.

ENERGY GAP OF A SEMICONDUCTOR DIODE

- 2.3 Determination of energy gap of a given semiconductor diode by measuring the variation of current as a function of temperature.
 - 2.4 Try to find the Fermi level of the given semiconductor
-

3. RESISTIVITY – FOUR PROBE METHOD

- 3.1 Determination of the resistivity by forcing current through two outer probes
 - 3.2 Formulate the reading of voltage across the two inner probes of semiconductor by four probe method.
-

4. MELDE'E EXPERIMENT

- 4.1 Determination of frequency of a given tuning fork in longitudinal wave propagation.
 - 4.2 Try to establish the transverse mode of wave propagation by understanding the theoretical harmonics and overtones.
-

5. B-H CURVE WITH CRO

- 5.1 Evaluate the energy loss per unit volume of a given magnetic material per cycle by tracing the hysteresis loop (B-H curve).
 - 5.2 Observe the hysteresis loss of ferro magnetic materials.
-

6. MAGNETIC MATERIAL

- 6.1 Determine the curie temperature (T_c) of a ferromagnetic material.
 - 6.2 Evaluate the relative permeability (μ_r) of a ferromagnetic material.
-

7. OPTICAL FIBER

- 7.1 Determine the numerical aperture of a given optical fiber.
 - 7.2 Calculate the acceptance angle of a given optical fiber.
-

8. LASER DIVERGENCE

- 8.1 Determination of the beam divergence of the given laser beam.
 - 8.2 Try to estimate the laser output
-

SOLAR CELL

- 8.3 Studying the characteristics of solar cell at different intensities
 - 8.4 Try to get the maximum workable power.
-

9. LIGHT EMITTING DIODE

- 9.1 Studying V-I characteristics of LED in forward bias for different LEDs.
 - 9.2 Measure the threshold voltage and forward resistance, and try for the dynamic Resistance
-

PLANCK'S CONSTANT

- 9.3 Determination of Planck's constant by measuring threshold voltage of given LED.
 - 9.4 Draw the L -I characteristics of the given LED.
-

10. STEWART GEE'S APPARATUS

- 10.1 Study the magnetic field along the axis of current carrying coil – Stewart and Gee's method.
- 10.2 Estimate the magnetic lines of force.

11. BIASING OF DIODE

- 11.1 Study the forward bias of LED
- 11.2 Study the reverse bias of Photo diode

V. TEXT BOOKS:

1. Laboratory Experiments in College Physics", C.H. Bernard and C.D. Epp, John Wiley and Sons, Inc., New York, 1995.

VI. REFERENCE BOOKS:

1. C. L. Arora, "Practical Physics", S. Chand & Co., New Delhi, 3rd Edition, 2012.
2. Vijay Kumar, Dr. T Radhakrishna, "Practical Physics for Engineering Students", SM Enterprises, 2nd Edition, 2014.
3. Dr. Rizwana, "Engineering Physics Manual", Spectrum Techno Press, 2018.

VII. ELECTRONICS RESOURCES:

4. <https://nptel.ac.in/translation>
5. <https://nptel.ac.in/courses/115105120>
6. NPTEL:: courses-Sem 1 and 2 - Engineering Physics and Applied Physics I
7. [Experimental Physics I - Course \(nptel.ac.in\)](https://nptel.ac.in)
8. [NPTEL:: Physics - Waves and Oscillations](https://nptel.ac.in)

VIII. MATERIALS ONLINE:

1. Course template
2. Lab manual

COURSE CONTENT

COMPUTER AIDED ENGINEERING DRAWING								
I Semester: AE / ME / CE								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AMED05	Foundation	L	T	P	C	CIA	SEE	Total
		0	1	2	2	40	60	100
Contact Classes: Nil	Tutorial Classes: 15	Practical Classes: 30			Total Classes: 45			
Prerequisite: There is no prerequisite required to this course								

I. COURSE OVERVIEW:

Engineering Drawing is the technique that develops the ability to visualize any object with all physical and dimensional configurations. The AutoCAD software assists in preparation of drawings to carry out sophisticated design and analysis of machine components and structures. This is the foundation course for civil engineering, mechanical engineering and aeronautical engineering that are improving their technologies in the era of digital manufacturing and construction.

II. COURSE OBJECTIVES:

The students will try to learn:

- The illustration of different objects using technical drawings using concepts of engineering drawing.
- The standard principles of orthographic projection of objects for making technical drawings.
- The representation of draw sectional views and pictorial views of solids.
- The computer aided drafting skills for producing the 2D and 3D drawings.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO1 Demonstrate the use of draw, modify and dimension commands of AutoCAD for development of drawings used in design and analysis of structures.
- CO2 Explain the constructional procedure of scales, conic sections and special curves used in engineering practices.
- CO3 Utilize the principles of orthographic projection for projections of points, lines, planes and regular solids using first angle projections.
- CO4 Interpret the sectional views and true shape of the section for revealing interior features of an object
- CO5 Illustrate the development of surfaces for construction of storage vessels, chemical vessels, boilers, and chimneys in industrial applications
- CO6 Make use of the concept of orthographic and isometric projections for converting isometric view to orthographic views and Vice-versa for engineering applications.

IV. COURE CONTENT:

MODULE – I: Introduction to engineering graphics

Principles of engineering graphics and their significance, scales, plain & diagonal, conic sections including the rectangular hyperbola, general method, cycloid, epicycloid and hypocycloid, introduction to computer aided drafting, views, commands.

MODULE – II: Orthographic projections

Principles of orthographic projections, conventions, projections of points and lines, projections of

plane regular geometric figures. Computer aided orthographic projections, points, lines and planes.

MODULE – III: Projections of regular solids

Projections of regular solids, auxiliary views, sections or sectional views of right regular solids, prism. Cylinder, pyramid, cone, computer aided projections of solids, sectional views.

MODULE – IV: Development of surfaces

Development of surfaces of right regular solids, prism, cylinder, pyramid and cone, development of surfaces using computer aided drafting.

MODULE – V: Isometric projections

principles of isometric projection, isometric scale, isometric views, conventions, isometric views of lines, plane figures, simple and compound solids, isometric projection of objects having non-isometric lines. Isometric projection of spherical parts, conversion of isometric views to orthographic views and vice-versa, conventions, conversion of orthographic projection into isometric view using computer aided drafting.

V.TEXT BOOKS:

1. N.D. Bhatt; *Engineering Drawing* Charotar Publishing House PVT Ltd, 15th edition 2011.
2. K. Venugopal; *Engineering Drawing and graphics Using AutoCAD*, 3rd edition 2007.

VI.REFERENCE BOOKS:

1. Basant Agrawal and C M Agrawal; *Engineering Drawing*, McGraw Hill, 3rd Edition 2011.
2. K L Narayana, P Kannaiah; *Engineering Drawing*, New age international (P) limited, 3rd edition, 2022.
3. M. B. Shah, B.C. Rane; *Engineering Drawing*, Pearson publications.

VII. ELECTRONIC RESOURCES:

1. <https://archive.nptel.ac.in/courses/112/103/112103019>.
2. <https://archive.nptel.ac.in/courses/112/105/112105294>.

VIII. MATERIALS ONLINE:

1. Course Template
2. Laboratory manual

EXERCISES COMPUTER AIDED ENGINEERING DRAWING

Note: Students are encouraged to bring their own laptops for laboratory practice sessions.

1. Getting Started Exercises

1.1 Introduction to Computer Aided Drafting (CAD)

- i) Engineering Drawing and its Significance
 - ii) Lettering and dimensioning
 - iii) Installation of AutoCAD
 - iv) Manufacturing of a product is the main activity in engineering profession. The design of a product may start with trial designs in the form of sketches on paper.
 - v) Advantages of CAD,
 - vi) Auto Cad Main Window
 - vii) The Coordinate System
 - viii) The Formats to Enter Coordinates
 - ix) Choosing Commands in AutoCAD
 - x) Object Snaps
 - xi) The Drawing Tools of CADD
-

2. Construction of scales

A scale is defined as the ratio of the linear dimensions of element of the object as represented in a drawing to the actual dimensions of the same element of the object itself.

It may not be always possible to prepare full-size drawings. They are, therefore, drawn proportionately smaller or larger. When drawings are drawn smaller than the actual size of the objects (as in case of buildings, bridges, large machines etc.) the scale used is said to be a reducing scale (example 1 : 5). Drawings of small machine parts, mathematical instruments, watches etc. are made larger than their real size. These are said to be drawn on an enlarging scale (example: 5 : 1).

Scales on drawings

To construct a scale the following information is essential:

- 1) The R.F. of the scale.
- 2) The units which it must represent, for example, millimetres and centimetres, or feet and inches etc.
- 3) The maximum length which it must show.

Types of Scales

The scales used in practice are classified as:

- 1) Plain scales
- 2) Diagonal scales
- 3) Vernier scales
- 4) Scale of chords.
- 5) Comparative scales

2.1 Exercises

i) Construction of plain scale

- 1. Construct a scale of 1:50 to read meters and decimeters and long enough to measure 6 m. Mark on it a distance of 5.5 m.
- 2. Construct a scale of 1.5 inches = 1 foot to show inches and long enough to measure upto 4 feet. Represent a distance of 2 feet and 10 inches on it.
- 3. Construct a scale of R.F.1/ 84480 to show miles and furlongs and long enough to measure up to 6 miles. Indicate on the scale a distance 3 miles and 4 furlongs.
- 4. The distance between Bombay and Poona is 180 km. A passenger train covers this distance in 6 hours.

Construct a plain scale to measure time upto a single minute. The R.F. or the scale is $1/2000$. Find the distance covered by the train in 36 minutes.

ii) Construction of diagonal scale

1. Construct a diagonal scale $1/50$, showing meters, decimeters and centimeters, to measure up to 5 meters. Mark a length 4.75 m on it.
2. An area of 144 sq cm on a map represents an area of 36 sq.km on the field. Find the RF of the scale of the map and draw a diagonal scale to show Km, hectometers and decameters and to measure up to 10 kilometers. Indicate on the scale a distance 7 kilometers, 5 hectometers and 6 decameters.
3. On a map, the distance between two points is 14 cm. The real distance between them is 20 km. Draw a diagonal scale of this map to read kilometers and hectometers, and to measure up to 25 km. Show a distance of 17.6 km on this scale.
4. Construct a diagonal scale of R.F = $1/4000$ and long enough to measure up to 500 meters. Also mark a distance of 374 m on it.

3. Construction of conic sections

The profile of number of objects consists of various types of curves. This chapter deals with various types of curves which are commonly used in engineering practice are: Conic sections, Evolutes, Cycloidal curves, Spirals, Involute and Helix.

3.1 Exercises

6. Construct a conic when the distance of its focus from the directrix is equal to 50 mm and its eccentricity is $3/4$. Draw a tangent at any point on the curve.
7. A fixed point is 75 mm from a fixed straight line. Draw the locus of a point P moving such a way that its distance from the fixed straight line is (i) twice its distance from the fixed point; (ii) equal to its distance from the fixed point. Name the curves.
8. Construct a hyperbola, when the distance of the focus from the directrix is 65 mm and eccentricity is $3/2$. Draw a tangent at any point on the curve.
9. Draw a parabola whose focus is at a distance of 75 mm from the directrix. Draw a tangent and normal at any point on it.
10. A point P is 30 mm and 50 mm respectively from two straight lines which are at right angles to each other. Draw a rectangular hyperbola from P within 10 mm distance from each line

4. Special curves used in engineering practices

4.1 Exercises

1. A circle of 50 mm diameter rolls along a straight line without slipping. Draw the curve traced out by a point P on the circumference, for one complete revolution of the circle. Name the curve. Draw a tangent to the curve at a point on it 40 mm from the line.
2. Circle of 50 mm diameter rolls on the circumference of another circle of 175 mm diameter and outside it. Trace the locus of a point on the circumference of the rolling circle for one complete revolution. Name the curve. Draw a tangent and a normal to the curve at a point 125 mm from the centre of the directing circle.
3. Construct a hypocycloid, rolling circle 50 mm diameter and directing circle 175 mm diameter. Draw a tangent to it at a point 50 mm from the centre of the directing circle.
4. A circle of 115 mm diameter rolls on another circle of 75 mm diameter with internal contact. Draw the locus of a point on the circumference of the rolling circle for its one complete revolution.
5. Draw an epicycloid having a generating circle of diameter 75 mm and a directing curve of radius 200 mm. Also draw a normal and a tangent at a point P on the curve.
6. Show by means of a drawing that when the diameter of the directing circle is twice that of the generating circle, the hypocycloid is a straight line. Take the diameter of the generating circle equal to 50 mm.

5. Orthographic projection

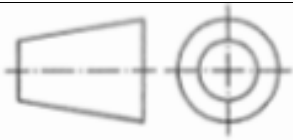
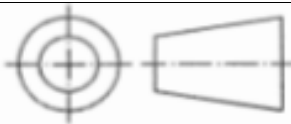
Practical solid geometry or descriptive geometry deals with the representation of points, lines, planes and solids on a flat surface (such as a sheet of paper), in such a manner that their relative positions and true forms can be accurately determined.

5.1 Types off projections

- I. Orthographic Projection
- II. Isometric Projection
- III. Oblique Projection
- IV. Perspective Projection

5.2 Symbol of Projections:

Table 5.1 Symbolic representation of first and third angle Projections

First Angle Projection		Third Angle Projection	
			
FV	LHSV	LHSV	FV

6 Projection of points

A point may be situated, in space, in any one of the four quadrants formed by the two principal planes of projection or may lie in any one or both of them. Its projections are obtained by extending projectors perpendicular to the planes.

One of the planes is then rotated so that the first and third quadrants are opened out as shown in figure 5.1. The projections are shown on a flat surface in their respective positions either above or below or in xy.

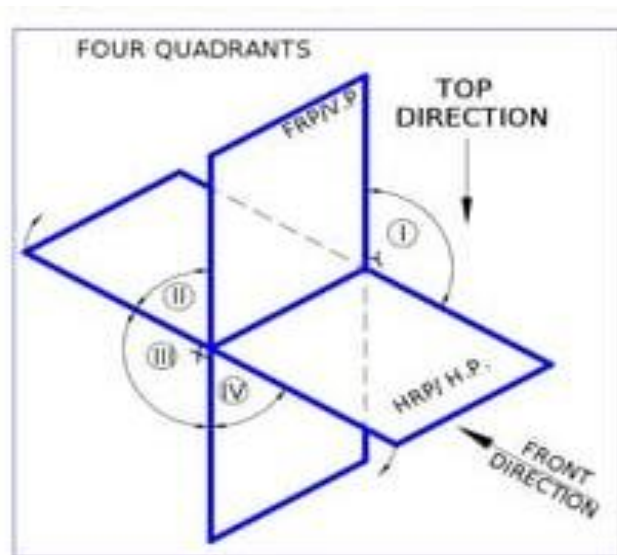


Figure 5.1: Four quadrants

6.1 Exercises

1. A point P is 15 mm above the H.P. and 20 mm in front of the V.P. Another point Q is 25 mm behind the V.P. and 40 mm below the H.P. Draw projections of P and Q keeping the distance between their projectors equal to 90 mm. Draw straight lines joining (i) their top views and (ii) their front views.
2. Two points A and B are in the H.P. The point A is 30 mm in front of the V.P., while B is behind the V.P. The distance between their projectors is 75 mm and the line joining their top views makes an angle of 45° with xy. Find the distance of the point B from the V.P.
3. A point A is situated in the first quadrant. Its shortest distance from the intersection point of H.P., V.P. and auxiliary plane is 60 mm and it is equidistant from the principal planes. Draw the projections of the point and determine its distance from the principal planes.
4. A point P is in the first quadrant. Its shortest distance from the intersection point of H.P., V.P. and Auxiliary vertical plane, perpendicular to the H.P. and V.P. is 70 mm and it is equidistant from principal planes (H.P. and V.P.). Draw the projections of the point and determine its distance from the H.P. and V.P.

7. Projection of lines

A straight line is the shortest distance between two points. Hence, the projections of a straight line may be drawn by joining the respective projections of its ends which are points. The position of a straight line may also be described with respect to the two reference planes. It may be:

1. Parallel to one or both the planes.
2. Contained by one or both the planes.
3. Perpendicular to one of the planes.
4. Inclined to one plane and parallel to the other.
5. Inclined to both the planes.
6. Projections of lines inclined to both the planes.
7. Line contained by a plane perpendicular to both the reference planes.
8. True length of a straight line and its inclinations with the reference planes.

7.1 Exercises

1. A line AB, 75 mm long, is inclined at 45° to the H.P. and 30° to the V.P. Its end B is in the H.P. and 40 mm in front of the V.P. Draw its projections
2. A line AB, 90 mm long is inclined at 30° to the H.P. Its end A is 12 mm above the H.P. and 20 mm in front of the V.P. Its front view measures 65 mm. Draw the top view of AB and determine its inclination with the V.P.
3. The top view of a 75 mm long line AB measures 65 mm, while the length of its front view is 50 mm. Its one end A is in the H.P. and 12 mm in front of the V.P. Draw the projections of AB and determine its inclinations with the H.P. and the V.P.
4. A line AB, 90 mm long, is inclined at 45° to the H.P. and its top view makes an angle of 60° with the V.P. The end A is in the H.P. and 12 mm in front of the V.P. Draw its front view and find its true inclination with the V.P.
5. A line PQ, 75 mm long, has its end P in the V.P. and the end Q in the H.P. The line is inclined at 30° to the H.P. and at 60° to the V.P. Draw its projections.

8. Projections of planes

Plane figures or surfaces have only two dimensions, viz. length and breadth. They do not have thickness. A plane figure may be assumed to be contained by a plane, and its projections can be drawn, if the position of that plane with respect to the principal planes of projection is known.

Types of planes:

Planes may be divided into two main types:

- **Perpendicular planes.**
 1. Perpendicular to both the reference planes.
 2. Perpendicular to one plane and parallel to the other.

3. Perpendicular to one plane and inclined to the other.

- **Oblique planes.**

Planes which are inclined to both the reference planes are called oblique planes.

8.1. Exercises

- 1) A square ABCD of 50 mm side has its corner A in the H.P., its diagonal AC inclined at 30° to the H.P. and the diagonal BD inclined at 45° to the V.P. and parallel to the H.P. Draw its projections.
- 2) Draw the projections of a rhombus having diagonals 125 mm and 50 mm long, the smaller diagonal of which is parallel to both the principal planes, while the other is inclined at 30° to the H.P.
- 3) Draw the projections of a regular pentagon of 40 mm side, having its surface inclined at 30° to the H.P. and a side parallel to the H.P. and inclined at an angle of 60° to the V.P.
- 4) Draw the projections of a regular hexagon of 40 mm side, having one of its sides in the H.P. and inclined at 60° to the V.P., and its surface making an angle of 45° with the H.P.
- 5) Draw the projections of a circle of 50 mm diameter resting in the H.P. on a point A on the circumference, its plane inclined at 45° to the H.P. and the top view of the diameter AB making 30° angle with the V.P.

9. Orthographic projections of solids

A solid has three dimensions, viz. length, breadth and thickness. For representing a solid on a flat surface having only length and breadth, at least two orthographic views are necessary. Sometimes, additional views projected on auxiliary planes become necessary to make the description of a solid complete.

9.1. Types of solids

Solid has three dimensions, the length, breadth and thickness or height. A solid may be represented by orthographic views, the number of which depends on the type of solid and its orientation with respect to the planes of projection. Solids are classified into two major groups. (i) Polyhedra, and (ii) Solids of revolution

Polyhedra

A polyhedra is defined as a solid bounded by plane surfaces called faces. They are:

- i. Regular polyhedral
- ii. Prisms
- iii. Pyramids.

Solids of Revolution

If a plane surface is revolved about one of its edges, the solid generated is called a solid of revolution. They are

- i. Cylinder
- ii. Cone
- iii. Sphere

9.2. Projections of solids in simple positions

- a. Axis perpendicular to the H.P.
- b. Axis perpendicular to the V.P.
- c. Axis parallel to both the H.P. and the V.P.

10. Projections of solids with axes inclined to one of the reference planes and parallel to the other

- a. Axis inclined to the V.P. and parallel to the H.P.
- b. Axis inclined to the H.P. and parallel to the V.P.

10.1 Exercises

- 1) A hexagonal prism, base 30 mm side and axis 75 mm long, has an edge of the base parallel to the H.P. and inclined at 45° to the V.P. Its axis makes an angle of 60° with the H.P. Draw its projections.
- 2) Draw the projections of a square pyramid having one of its triangular faces in the V.P. and the axis parallel to and 40 mm above the H.P. Base 30 mm side; axis 75 mm long.

- 3) Draw the projections of a cone, base 45 mm diameter and axis 50 mm long, when it is resting on the ground on a point on its base circle with the axis making an angle of 30° with the H.P. and its top view making 45° with the V.P.
- 4) A pentagonal pyramid, base 30 mm side and axis 65 mm long has one of its triangular faces in the V.P. and the edge of the base contained by that face makes an angle of 30° with the H.P. Draw its projections.
- 5) Draw the projections of a cone, base 50 mm diameter and axis 75 mm long, lying on a generator on the ground with the top view of the axis making an angle of 45° with the V.P.

11. Section of Solids

Invisible features of an object are shown by dotted lines in their projected views. But when such features are too many, these lines make the views more complicated and difficult to interpret. In such cases, it is customary to imagine the object as being cut through or sectioned by planes. The part of the object between the cutting plane and the observer is assumed to be removed and the view is then shown in section.

- The imaginary plane is called a section plane or a cutting plane.
- The surface produced by cutting the object by the section plane is called the section. It is indicated by thin section lines uniformly spaced and inclined at 45° .
- The projection of the section on a plane parallel to the section plane will show the true shape of the section is called True shape of the section.

11.1 Exercises

- 1) A pentagonal pyramid, base 30 mm side and axis 65 mm long has its base horizontal and an edge of the base parallel to the V.P. A horizontal section plane cuts it at a distance of 25 mm above the base. Draw its front view and sectional top view.
- 2) A cylinder of 40 mm diameter, 60 mm height and having its axis vertical, is cut by a section plane, perpendicular to the V.P., inclined at 45° to the H.P. and intersecting the axis 32 mm above the base. Draw its front view, sectional top view, sectional side view and true shape of the section.
- 3) A cone, base 75 mm diameter and axis 80 mm long is resting on its base on the H.P. It is cut by a section plane perpendicular to the VP, inclined at 45° to the H.P. and cutting the axis at a point 35 mm from the apex. Draw its front view, sectional top view, sectional side view and true shape of the section.
- 4) A hexagonal prism, side of base 35 mm and height 75 mm is resting on one of its corners on the H.P. with a longer edge containing that corner inclined at 60° to the H.P. and a rectangular face parallel to the V.P. A horizontal section plane cuts the prism in two equal halves. Draw the front view and sectional top view of the cut prism.

12. Development of surfaces

Imagine that a solid is enclosed in a wrapper of thin material, such as paper. If this covering is opened out and laid on a flat plane, the flattened-out paper is the development of the solid. Thus, when surfaces of a solid are laid out on a plane, the figure obtained is called its development.

12.1 Methods of development.

1. Parallel-line: It is employed in case of prisms and cylinders in which stretch-out-line principle is used.
2. Radial line: It is used for pyramids and cones in which the true length of the slant edge or the generator is used as radius.
3. Triangulation: This is used to develop transition pieces. This is simply a method of dividing a surface into a number of triangles and transferring them into the development.
4. Approximate: It is used to develop objects of double curved or warped surfaces as sphere, paraboloid, ellipsoid, hyperboloid and helicoids.

12.2 Exercise

1. A hexagonal prism of side of base 30 mm and axis 70 mm long is resting on its base on HP. such that a rectangular face is parallel to V.P. It is cut by a section plane perpendicular to VP and inclined at 30° to HP. The section plane is passing through the top end of an extreme lateral edge of the prism. Draw the development of the lateral surface of the cut prism.
2. A pentagonal prism with edge of base 30 mm and height 80 mm rests on its base with one of its base edges perpendicular to VP. An inclined plane at 45° to H.P. cuts its axis at its middle. Draw the development of the truncated prism
3. A cylinder of diameter of base 40 mm and height 50 mm is standing on its base on HP. A cutting plane inclined at 45° to the axis of the cylinder passes through the left extreme point of the top base. Develop the lateral surface of the truncated cylinder.
4. A hexagonal pyramid with side of base 30 mm and height 75 mm stands with its base on HP and an edge of the base parallel to VP. It is cut by a plane perpendicular to VP, inclined at 45° to H.P and passing through the mid-point of the axis. Draw the (sectioned) top view and develop the lateral surface of the truncated pyramid
5. A pentagonal pyramid, side of base 50 mm and height 80 mm rests on its base on the ground with one of its base sides parallel to V.P. A section plane perpendicular to VP and inclined at 30° to H.P cuts the pyramid, bisecting its axis. Draw the development of the truncated pyramid
6. A cone of diameter of base 75 mm and height 60 mm is cut by horizontal cutting plane at 20 mm from the apex. Draw the development of the truncated cone.
7. A cone of base 50mm diameter and height 60 mm rests with its base on H.P. and bisects the axis of the cone. Draw the development of the lateral surface of the truncated cone.

13 Isometric Views

Isometric projection is a type of pictorial projection in which the three dimensions of a solid are not only shown in one view, but their actual sizes can be measured directly from it.

13.1 Exercises

1. Draw the isometric view, Principle of isometric Projections and isometric scale
2. Draw the isometric view of square, pentagon and hexagon of side length 40 mm, kept (i) parallel to VP and (ii) parallel to HP.
3. Draw the isometric view of circle of 60 mm diameter kept (i) parallel to VP and (ii) parallel to HP
4. Draw the isometric drawing of a cone of base diameter 30mm and axis 50 mm long resting on its base on HP.
5. A pentagonal pyramid of side of base 30 mm and height 70 mm is resting with its base on HP and one of its edges is parallel to VP. Draw the isometric drawing of the pyramid.

13.2 Draw the isometric views of given objects

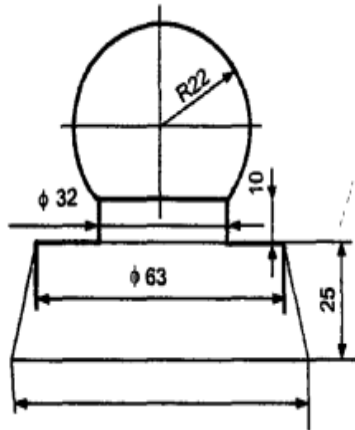


Figure 13.1

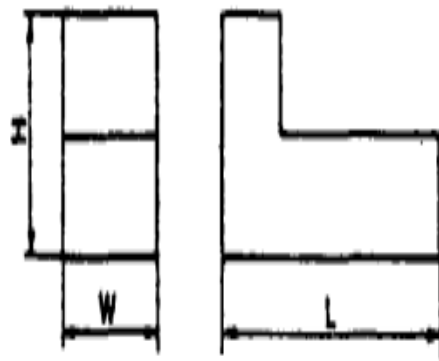


Figure 13.2

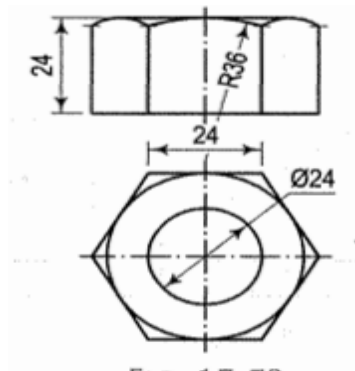


Figure 13.3

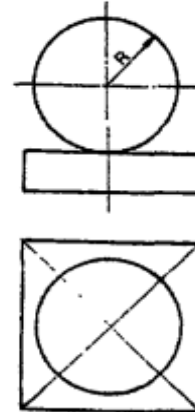


Figure 13.4

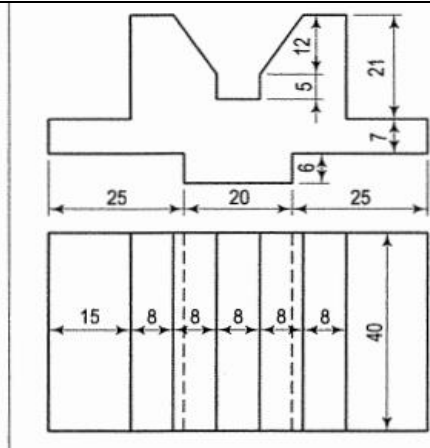


Figure 13.5

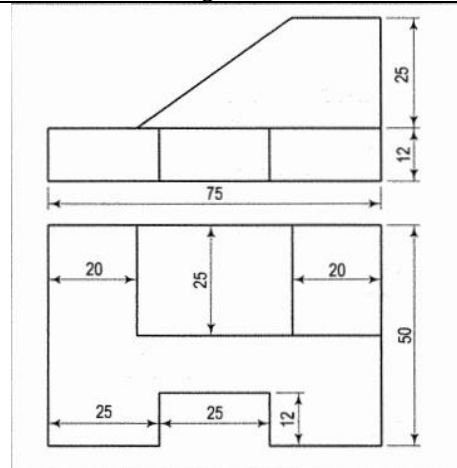
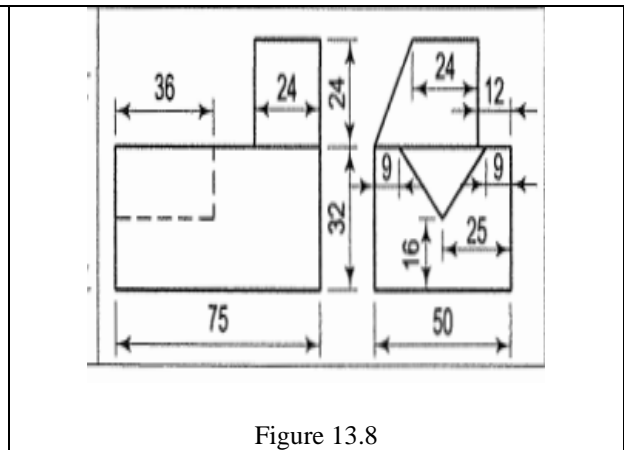
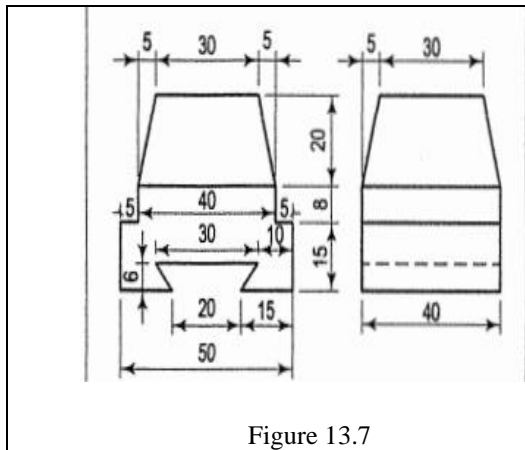


Figure 13.6

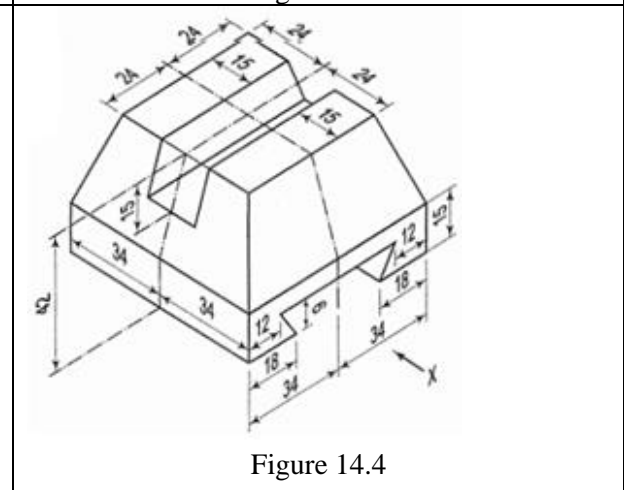
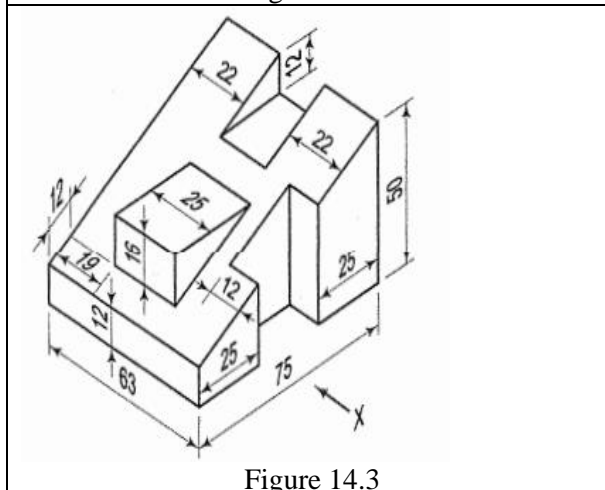
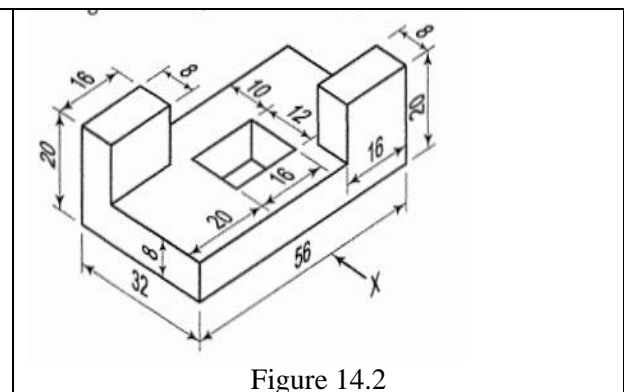
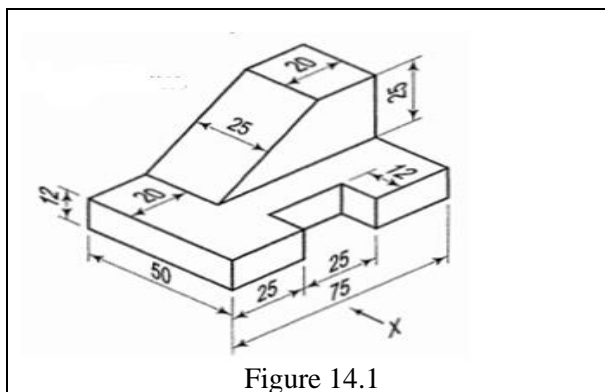


14 Pictorial views into Orthographic Projections

Orthographic reading is the ability to visualize the shape of an object from its drawing in orthographic views. Every engineer or technician connected with the work of construction should possess this ability. Without it, it would be difficult to execute, independently, any work according to a given drawing.

14.1 Exercise

Using first-angle projection method, draw the front view, top view Insert and side of a given pictorial view. Insert all dimensions in the views.





INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

PROGRAMMING FOR PROBLEM SOLVING LABORATORY								
II Semester: AE / ME / CE / ECE / EEE / CSE / CSE (AI & ML) / CSE (DS) / CSE (CS) / IT								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
ACSD06	Foundation	L	T	P	C	CIA	SEE	Total
		0	1	2	2	40	60	100
Contact Classes: Nil	Tutorial Classes: 15	Practical Classes: 30			Total Classes: 45			
Prerequisites: There are no prerequisites to take this course.								

I. COURSE OVERVIEW:

The course is designed with the fundamental programming skills and problem-solving strategies necessary to tackle a wide range of computational challenges. Through hands-on programming exercises and projects, students will learn how to write code, analyze problems and develop solutions using various programming languages and tools. The course will cover fundamental programming concepts and gradually progress to more advanced topics.

II. COURSE OBJECTIVES

The students will try to learn:

- I. The fundamental programming constructs and use of collection data types in Python.
- II. The ability to develop programs using object-oriented features.
- III. Basic data structures and algorithms for efficient problem-solving.
- IV. Principles of graph theory and be able to apply their knowledge to a wide range of practical problems across various disciplines.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO1 Adapt programming concepts, syntax, and data structures through hands on coding exercises.
- CO2 Develop the ability to solve a variety of programming problems and algorithms using python.
- CO3 Implement complex and custom data structures to solve real-world problems.
- CO4 Demonstrate proficiency in implementing graph algorithms to solve variety of problems and scenarios.
- CO5 Develop critical thinking skills to solve the various real-world applications using graph theory.
- CO6 Learn the importance of numerical methods and apply them to tackle a wide range of computational problems.

IV. COURSE CONTENT:

EXERCISES FOR PROGRAMMING FOR PROBLEM SOLVING LABORATORY

Note: Students are encouraged to bring their own laptops for laboratory practice sessions.

1. Getting Started Exercises

1.1 Two Sum

Given an array of integers `nums` and an integer `target`, return indices of the two numbers such that they add up to `target`. You may assume that each input would have exactly one solution, and you may not use the same element twice. You can return the answer in any order.

Input: `nums = [2, 7, 11, 15]`, `target = 9`

Output: `[0, 1]`

Explanation: Because `nums[0] + nums[1] == 9`, so return `[0, 1]`.

Input: `nums = [3, 2, 4]`, `target = 6`

Output: `[1, 2]`

Input: `nums = [3, 3]`, `target = 6`

Output: `[0, 1]`

Hints:

```
def twoSum(self, nums: List[int], target: int) -> List[int]:
    a=[]
    # Write code here
    ...

    return a
```

1.2 Contains Duplicate

Given an integer array `nums`, return `true` if any value appears at least twice in the array, and return `false` if every element is distinct.

Input: `nums = [1, 2, 3, 1]`

Output: `true`

Input: `nums = [1, 2, 3, 4]`

Output: `false`

Input: `nums = [1, 1, 1, 3, 3, 4, 3, 2, 4, 2]`

Output: `true`

Hints:

```
def containsDuplicate(self, nums):
    a = set() # set can have only distinct elements
    # Write code here
    ...

    return False
```

1.3 Roman to Integer

Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M.

Symbol	Value
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

For example, 2 is written as II in Roman numeral, just two ones added together. 12 is written as XII, which is simply X + II. The number 27 is written as XXVII, which is XX + V + II.

Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not IIII. Instead, the number four is written as IV. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as IX. There are six instances where subtraction is used:

I can be placed before V (5) and X (10) to make 4 and 9.

X can be placed before L (50) and C (100) to make 40 and 90.

C can be placed before D (500) and M (1000) to make 400 and 900.

Given a roman numeral, convert it to an integer.

Input: s = "III"

Output: 3

Input: s = "LVIII"

Output: 58

Hints:

```
def romanToInt(self, s: str) -> int:
    # Write code here
    ...

    return number
```

1.4 Plus One

You are given a large integer represented as an integer array digits, where each digits[i] is the ith digit of the integer. The digits are ordered from most significant to least significant in left-to-right order. The large integer does not contain any leading 0's. Increment the large integer by one and return the resulting array of digits.

Input: digits = [1, 2, 3]

Output: [1, 2, 4]

Explanation: The array represents the integer 123.

Incrementing by one gives 123 + 1 = 124.

Thus, the result should be [1, 2, 4].

Hints:

```
def plusOne(self, digits: List[int]) -> List[int]:
    n = len(digits)
    # Write code here
    ...

    return digits
```

1.5 Majority Element

Given an array nums of size n, return the majority element. The majority element is the element that appears more than $\lfloor n / 2 \rfloor$ times. You may assume that the majority element always exists in the array.

Input: nums = [3, 2, 3]

Output: 3

Input: nums = [2, 2, 1, 1, 1, 2, 2]

Output: 2

Hints:

```
def majorityElement(self, nums):
    # write code here
    ...
```

1.6 Richest Customer Wealth

You are given an $m \times n$ integer grid `accounts` where `accounts[i][j]` is the amount of money the i^{th} customer has in the j^{th} bank. Return the wealth that the richest customer has. A customer's wealth is the amount of money they have in all their bank accounts. The richest customer is the customer that has the maximum wealth.

Input: `accounts = [[1, 2, 3], [3, 2, 1]]`

Output: 6

Explanation:

1st customer has wealth = $1 + 2 + 3 = 6$

2nd customer has wealth = $3 + 2 + 1 = 6$

Both customers are considered the richest with a wealth of 6 each, so return 6.

Input: `accounts = [[1, 5], [7, 3], [3, 5]]`

Output: 10

Explanation:

1st customer has wealth = 6

2nd customer has wealth = 10

3rd customer has wealth = 8

The 2nd customer is the richest with a wealth of 10.

Input: `accounts = [[2, 8, 7], [7, 1, 3], [1, 9, 5]]`

Output: 17

Hints:

```
def maximumWealth(self, accounts: List[List[int]]) -> int:
```

```
    # write code here
```

```
    ...
```

1.7 Fizz Buzz

Given an integer n , return a string array `answer` (1-indexed) where:

`answer[i] == "FizzBuzz"` if i is divisible by 3 and 5.

`answer[i] == "Fizz"` if i is divisible by 3.

`answer[i] == "Buzz"` if i is divisible by 5.

`answer[i] == i` (as a string) if none of the above conditions are true.

Input: $n = 3$

Output: `["1", "2", "Fizz"]`

Input: $n = 5$

Output: `["1", "2", "Fizz", "4", "Buzz"]`

Input: $n = 15$

Output: `["1", "2", "Fizz", "4", "Buzz", "Fizz", "7", "8", "Fizz", "Buzz", "11", "Fizz", "13", "14", "FizzBuzz"]`

Hints:

```
def fizzBuzz(self, n: int) -> List[str]:
```

```
    # write code here
```

```
    ...
```

1.8 Number of Steps to Reduce a Number to Zero

Given an integer `num`, return the number of steps to reduce it to zero. In one step, if the current number is even, you have to divide it by 2, otherwise, you have to subtract 1 from it.

Input: `num = 14`

Output: 6

Explanation:

- 14 is even; divide by 2 and obtain 7.
- 7 is odd; subtract 1 and obtain 6.
- 6 is even; divide by 2 and obtain 3.
- 3 is odd; subtract 1 and obtain 2.
- 2 is even; divide by 2 and obtain 1.
- 1 is odd; subtract 1 and obtain 0.

Input: num = 8

Output: 4

Explanation:

- 8 is even; divide by 2 and obtain 4.
- 4 is even; divide by 2 and obtain 2.
- 2 is even; divide by 2 and obtain 1.
- 1 is odd; subtract 1 and obtain 0.

Input: num = 123

Output: 12

Hints:

```
def numberOfSteps(self, n: int) -> int:
    # write code here
    ...
```

1.9 Running Sum of 1D Array

Given an array nums. We define a running sum of an array as $\text{runningSum}[i] = \text{sum}(\text{nums}[0] \dots \text{nums}[i])$.

Return the running sum of nums.

Input: nums = [1, 2, 3, 4]

Output: [1, 3, 6, 10]

Explanation: Running sum is obtained as follows: [1, 1+2, 1+2+3, 1+2+3+4].

Input: nums = [1, 1, 1, 1, 1]

Output: [1, 2, 3, 4, 5]

Explanation: Running sum is obtained as follows: [1, 1+1, 1+1+1, 1+1+1+1, 1+1+1+1+1].

Input: nums = [3, 1, 2, 10, 1]

Output: [3, 4, 6, 16, 17]

Hints:

```
def runningSum(self, nums: List[int]) -> List[int]:
    # write code here
    ...
    return answer
```

1.10 Remove Element

Given an integer array nums and an integer val, remove all occurrences of val in nums in-place. The order of the elements may be changed. Then return the number of elements in nums which are not equal to val. Consider the number of elements in nums which are not equal to val be k, to get accepted, you need to do the following things:

- Change the array nums such that the first k elements of nums contain the elements which are not equal to val. The remaining elements of nums are not important as well as the size of nums.
- Return k.

Input: nums = [3, 2, 2, 3], val = 3

Output: 2, nums = [2, 2, _, _]

Explanation: Your function should return k = 2, with the first two elements of nums being 2.

It does not matter what you leave beyond the returned k (hence they are underscores).

Input: nums = [0,1,2,2,3,0,4,2], val = 2

Output: 5, nums = [0,1,4,0,3,_,_,_]

Explanation: Your function should return k = 5, with the first five elements of nums containing 0, 0, 1, 3, and 4.

Note that the five elements can be returned in any order.

It does not matter what you leave beyond the returned k (hence they are underscores).

Hints:

```
def removeElement(self, nums: List[int], val: int) -> int:
    # write code here
    ...
    return len(nums)
```

2. Matrix Operations

2.1 Add Two Matrices

Given two matrices X and Y, the task is to compute the sum of two matrices and then print it in Python.

Input:

```
X= [[1, 2, 3],
     [4, 5, 6],
     [7, 8, 9]]
```

```
Y = [[9, 8, 7],
     [6, 5, 4],
     [3, 2, 1]]
```

Output:

```
Result = [[10, 10, 10],
          [10, 10, 10],
          [10, 10, 10]]
```

Hints:

```
# Program to add two matrices using nested loop
```

```
X = [[1, 2, 3],
     [4, 5, 6],
     [7, 8, 9]]
```

```
Y = [[9,8,7],
     [6,5,4],
     [3,2,1]]
```

```
result = [[0,0,0],
          [0,0,0],
          [0,0,0]]
```

```
# iterate through rows
for i in range(len(X)):
    # write code here
    ...
```

```
for r in result:
```

```
print(r)
```

TRY

1. Take input as X = [[10, 20, 30],[41, 52, 63], [47, 58, 69]] Y = [[19,18,17],[66,35,49], [13,21,11]] and verify the results.

2.2 Multiply Two Matrices

Given two matrices X and Y, the task is to compute the multiplication of two matrices and then print it.

Input:

```
X= [[1, 7, 3],  
    [3, 5, 6],  
    [6, 8, 9]]
```

```
Y = [[1, 1, 1, 2],  
     [6, 7, 3, 0],  
     [4, 5, 9, 1]]
```

Output:

```
Result = [[55, 65, 49, 5],  
          [57, 68, 72, 12],  
          [90, 107, 111, 21]]
```

Hints:

```
# Program to multiply two matrices using list comprehension
```

```
# take a 3x3 matrix
```

```
A = [[12, 7, 3],  
     [4, 5, 6],  
     [7, 8, 9]]
```

```
# take a 3x4 matrix
```

```
B = [[5, 8, 1, 2],  
     [6, 7, 3, 0],  
     [4, 5, 9, 1]]
```

```
# result will be 3x4
```

```
# write code here
```

```
...  
for r in result:  
    print(r)
```

TRY

1. Take input as X = [[11, 0, 30],[-41, -2, 63], [41, -5, -9]] Y = [[19,-48,17],[-6,35,19], [13,1,-9]] and verify the results.

2.3 Transpose of a Matrix

A matrix can be implemented using a nested list. Each element is treated as a row of the matrix. Find the transpose of a matrix in multiple ways.

Input: [[1, 2], [3, 4], [5, 6]]

Output: [[1, 3, 5], [2, 4, 6]]

Explanation: Suppose we are given a matrix

```
[[1, 2],  
 [3, 4],  
 [5, 6]]
```

Then the transpose of the given matrix will be,

```
[[1, 3, 5],  
 [2, 4, 6]]
```


Hints:

```
# Program to multiply two matrices using list comprehension
```

```
# take a 3x3 matrix
```

```
A = [[12, 7, 3],
      [4, 5, 6],
      [7, 8, 9]]
```

```
# result will be 3x4
```

```
# write code here
```

```
...
for r in result:
    print(r)
```

TRY

1. Take input as X = [[11, 0, 30], [-41, -2, 63], [41, -5, -9]] and verify the results.

2.4 Matrix Product

Matrix product problem we can solve using list comprehension as a potential shorthand to the conventional loops. Iterate and find the product of the nested list and at the end return the cumulative product using function.

Input: The original list: [[1, 4, 5], [7, 3], [4], [46, 7, 3]]

Output: The total element product in lists is: 1622880

Hints:

```
# Matrix Product using list comprehension + loop
```

```
def prod(val):
    # write code here
    ...
```

```
# initializing list
```

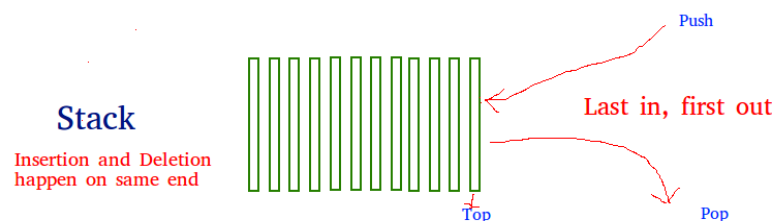
```
test_list = [[1, 4, 5], [7, 3], [4], [46, 7, 3]]
```

TRY

1. Take input list: [[1, 4, 5], [7, 3], [4], [46, 7, 3]] and verify the result.

3. Stack**3.1 Stack implementation using List**

A stack is a linear data structure that stores items in a Last-In/First-Out (LIFO) or First-In/Last-Out (FILO) manner. In stack, a new element is added at one end and an element is removed from that end only. The insert and delete operations are often called push and pop.



The functions associated with stack are:

- **empty()** – Returns whether the stack is empty
- **size()** – Returns the size of the stack
- **top() / peek()** – Returns a reference to the topmost element of the stack
- **push(a)** – Inserts the element 'a' at the top of the stack
- **pop()** – Deletes the topmost element of the stack

Hints:

```
# Stack implementation using list
```

```
top=0
```

```
mymax=5
```

```
def createStack():
```

```
    stack=[]
```

```
    return stack
```

```
def isEmpty(stack):
```

```
    # write code here
```

```
    ...
```

```
def Push(stack,item):
```

```
    # write code here
```

```
    ...
```

```
def Pop(stack):
```

```
    # write code here
```

```
    ...
```

```
# create a stack object
```

```
stack = createStack()
```

```
while True:
```

```
    print("1.Push")
```

```
    print("2.Pop")
```

```
    print("3.Display")
```

```
    print("4.Quit")
```

```
    # write code here
```

```
    ...
```

TRY

1. Take input operations as [PUSH(A),PUSH(B),PUSH(C),POP,POP,POP,POP,PUSH(D)] and verify the result.
2. Take input operations as [POP, POP, PUSH (A), PUSH (B), POP, and PUSH(C)] and verify the result.

3.2 Balanced Parenthesis Checking

Given an expression string, write a python program to find whether a given string has balanced parentheses or not.

Input: {[]{}() }

Output: Balanced

Input: [{ } { } (]

Output: Unbalanced

Using stack One approach to check balanced parentheses is to use stack. Each time, when an open parentheses is encountered push it in the stack, and when closed parenthesis is encountered, match it with the top of stack and pop it. If stack is empty at the end, return Balanced otherwise, Unbalanced.

Hints:

```
# Check for balanced parentheses in an expression
```

```
open_list = ["[","{","("]
```

```
close_list = ["]","}",")"]
```

```
# Function to check parentheses
```

```
def check(myStr):
```

```
    # write code here    ...
```

TRY

1. Take input as $\{\{\}\{\}\{\}\{\}\}$ and verify the result.
2. Take input as $\{\{\}\{\}\{\}\{\}\{\}\}$ and verify the result.

3.3 Evaluation of Postfix Expression

Given a postfix expression, the task is to evaluate the postfix expression. Postfix expression: The expression of the form “a b operator” (ab+) i.e., when a pair of operands is followed by an operator.

Input: str = “2 3 1 * + 9 -“

Output: -4

Explanation: If the expression is converted into an infix expression, it will be $2 + (3 * 1) - 9 = 5 - 9 = -4$.

Input: str = “100 200 + 2 / 5 * 7 +”

Output: 757

Procedure for evaluation postfix expression using stack:

- Create a stack to store operands (or values).
- Scan the given expression from left to right and do the following for every scanned element.
 - If the element is a number, push it into the stack.
 - If the element is an operator, pop operands for the operator from the stack. Evaluate the operator and push the result back to the stack.
- When the expression is ended, the number in the stack is the final answer.

Hints:

Evaluate value of a postfix expression

Class to convert the expression

class Evaluate:

Constructor to initialize the class variables

```
def __init__(self, capacity):
    self.top = -1
    self.capacity = capacity
```

```
# This array is used as a stack
self.array = []
```

Check if the stack is empty

```
def isEmpty(self):
    # write code here
    ...
```

```
def peek(self):
    # write code here
    ...
```

```
def pop(self):
    # write code here
    ...
```

```
def push(self, op):
    # write code here
    ...
```

```
def evaluatePostfix(self, exp):
    # write code here
    ...
```

```
# Driver code
exp = "231*+9-"
obj = Evaluate(len(exp))

# Function call
print("postfix evaluation: %d" % (obj.evaluatePostfix(exp)))
```

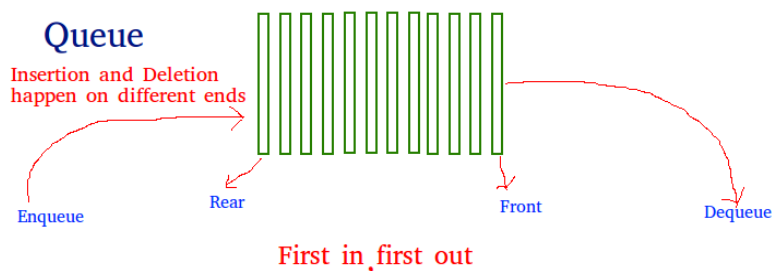
TRY

1. Take input str = "A B + C* D / E +" and verify the result.
2. Take input str = "XYZ- + W+ R / S -" and verify the result.

4. Queue

4.1 Linear Queue using List

Linear queue is a linear data structure that stores items in First in First out (FIFO) manner. With a queue the least recently added item is removed first. A good example of queue is any queue of consumers for a resource where the consumer that came first is served first.



Hints:

```
# Static implementation of linear queue
front=0
rear=0
mymax=5
def createQueue():
    queue=[] #empty list
    return queue

def isEmpty(queue):
    # write code here
    ...

def enqueue(queue,item): # insert an element into the queue
    # write code here
    ...

def dequeue(queue): #remove an element from the queue
    # write code here
    ...

# Driver code
queue = createQueue()
while True:
    print("1.Enqueue")
    print("2.Dequeue")
    print("3.Display")
```

```
print("4.Quit")
# write code here
...
```

TRY

1. Take input operations as [ENQUEUE(A),DEQUEUE(),ENQUEUE(B),DEQUEUE(),ENQUEUE(C),DEQUEUE(),] and verify the result.
2. Take input operations as [ENQUEUE(A), ENQUEUE(B),DEQUEUE(),ENQUEUE(C), DEQUEUE(),ENQUEUE(D), DEQUEUE(), ENQUEUE(C),DEQUEUE(),] and verify the result.

4.2 Stack using Queues

Implement a last-in-first-out (LIFO) stack using only two queues. The implemented stack should support all the functions of a normal stack (push, top, pop, and empty).

- void push(int x) Pushes element x to the top of the stack.
- int pop() Removes the element on the top of the stack and returns it.
- int top() Returns the element on the top of the stack.
- boolean empty() Returns true if the stack is empty, false otherwise.

Input:

```
["MyStack", "push", "push", "top", "pop", "empty"]
[[], [1], [2], [], [], []]
```

Output:

```
[null, null, null, 2, 2, false]
```

Hints:

```
class MyStack:

    def __init__(self):
        # write code here
        ...

    def push(self, x: int) -> None:
        # write code here
        ...

    def pop(self) -> int:
        # write code here
        ...

    def top(self) -> int:
        # write code here
        ...

    def empty(self) -> bool:
        # write code here
        ...

# Your MyStack object will be instantiated and called as such:
# obj = MyStack()
# obj.push(x)
# param_2 = obj.pop()
# param_3 = obj.top()
# param_4 = obj.empty()
```

4.3 Implement Queue using Stacks

Implement a first in first out (FIFO) queue using only two stacks. The implemented queue should support all the functions of a normal queue (push, peek, pop, and empty).

- void push(int x) Pushes element x to the back of the queue.
- int pop() Removes the element from the front of the queue and returns it.
- int peek() Returns the element at the front of the queue.
- boolean empty() Returns true if the queue is empty, false otherwise.

Input:

["MyQueue", "push", "push", "peek", "pop", "empty"]

[[], [1], [2], [], [], []]

Output:

[null, null, null, 1, 1, false]

Hints:

```
class MyQueue:

    def __init__(self):
        # write code here
        ...

    def push(self, x: int) -> None:
        # write code here
        ...

    def pop(self) -> int:
        # write code here
        ...

    def peek(self) -> int:
        # write code here
        ...

    def empty(self) -> bool:
        # write code here
        ...

# Your MyQueue object will be instantiated and called as such:
# obj = MyQueue()
# obj.push(x)
# param_2 = obj.pop()
# param_3 = obj.peek()
# param_4 = obj.empty()
```

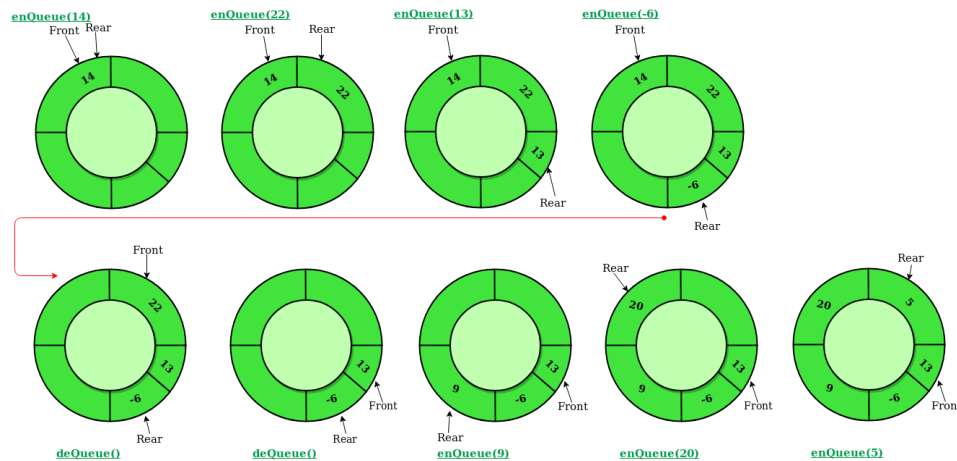
4.4 Circular Queue

A Circular Queue is an extended version of a normal queue where the last element of the queue is connected to the first element of the queue forming a circle. The operations are performed based on FIFO (First In First Out) principle. It is also called 'Ring Buffer'.

Operations on Circular Queue:

- **Front:** Get the front item from the queue.
- **Rear:** Get the last item from the queue.
- **enQueue(value)** This function is used to insert an element into the circular queue. In a circular queue, the new element is always inserted at the rear position.

- Check whether the queue is full – [i.e., the rear end is in just before the front end in a circular manner].
- If it is full then display Queue is full.
- If the queue is not full then, insert an element at the end of the queue.
- **deQueue()** This function is used to delete an element from the circular queue. In a circular queue, the element is always deleted from the front position.
- Check whether the queue is Empty.
- If it is empty then display Queue is empty.
- If the queue is not empty, then get the last element and remove it from the queue.



Implement Circular Queue using Array:

1. Initialize an array queue of size **n**, where **n** is the maximum number of elements that the queue can hold.
2. Initialize two variables front and rear to -1.
3. **Enqueue:** To enqueue an element **x** into the queue, do the following:
 - Increment rear by 1.
 - If **rear** is equal to **n**, set **rear** to 0.
 - If **front** is -1, set **front** to 0.
 - Set queue[rear] to **x**.
4. **Dequeue:** To dequeue an element from the queue, do the following:
 - Check if the queue is empty by checking if **front** is -1.
 - If it is, return an error message indicating that the queue is empty.
 - Set **x** to queue [front].
 - If **front** is equal to **rear**, set **front** and **rear** to -1.
 - Otherwise, increment **front** by 1 and if **front** is equal to **n**, set **front** to 0.
 - Return **x**.

Hints:

```
class CircularQueue():

    # constructor
    def __init__(self, size): # initializing the class
        self.size = size

    # initializing queue with none
    self.queue = [None for i in range(size)]
    self.front = self.rear = -1

    def enqueue(self, data):
        # Write code here
        ...

    def dequeue(self):
        # Write code here
        ...
```

```

def display(self):
    # Write code here
    ...

# Driver Code
ob = CircularQueue(5)
ob.enqueue(14)
ob.enqueue(22)
ob.enqueue(13)
ob.enqueue(-6)
ob.display()
print ("Deleted value = ", ob.dequeue())
print ("Deleted value = ", ob.dequeue())
ob.display()
ob.enqueue(9)
ob.enqueue(20)
ob.enqueue(5)
ob.display()

```

TRY

1. Take input operations as [ENQUEUE(A,B,C,D,E,F),DEQUEUE(),DEQUEUE(),DEQUEUE(),ENQUEUE(G,H,I)] and verify the result.
2. Take input operations as [DEQUEUE(),ENQUEUE(A,B,C,D,E,F),DEQUEUE(),ENQUEUE(G,H,I)] and verify the result.

5. Graph Representation

5.1 Build a graph

You are given an integer n . Determine if there is an unconnected graph with n vertices that contains at least two connected components and contains the number of edges that is equal to the number of vertices. Each vertex must follow one of these conditions:

- Its degree is less than or equal to 1.
- It's a cut-vertex.

Note:

- The graph must be simple.
- Loops and multiple edges are not allowed.

Input: First line: n

Output: Print Yes if it is an unconnected graph. Otherwise, print No.

Sample Input	Sample Output
3	No

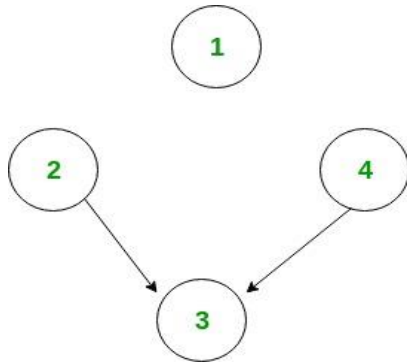
Constraints: $1 \leq n \leq 100$

Explanation: There is only one graph with the number of edges equal to the number of vertices (triangle) which is connected.

5.2 Number of Sink Nodes in a Graph

Given a Directed Acyclic Graph of n nodes (numbered from 1 to n) and m edges. The task is to find the number of sink nodes. A sink node is a node such that no edge emerges out of it.

Input: $n = 4$, $m = 2$, edges[] = {{2, 3}, {4, 3}}



Only node 1 and node 3 are sink nodes.

Input: $n = 4$, $m = 2$, $\text{edges[]} = \{\{3, 2\}, \{3, 4\}\}$

Output: 3

The idea is to iterate through all the edges. And for each edge, mark the source node from which the edge emerged out. Now, for each node check if it is marked or not. And count the unmarked nodes.

Algorithm:

1. Make any array $A[]$ of size equal to the number of nodes and initialize to 1.
2. Traverse all the edges one by one, say, $u \rightarrow v$.
 - (i) Mark $A[u]$ as 1.
3. Now traverse whole array $A[]$ and count number of unmarked nodes.

Hints:

```

# Program to count number if sink nodes

# Return the number of Sink Nodes.
def countSink(n, m, edgeFrom, edgeTo):
    # Write code here
    ...

    return count

# Driver Code
n = 4
m = 2
edgeFrom = [2, 4]
edgeTo = [3, 3]

print(countSink(n, m, edgeFrom, edgeTo))

```

5.3 Connected Components in a Graph

Given n , i.e. total number of nodes in an undirected graph numbered from 1 to n and an integer e , i.e. total number of edges in the graph. Calculate the total number of connected components in the graph. A connected component is a set of vertices in a graph that are linked to each other by paths.

Input: First line of input line contains two integers' n and e . Next e line will contain two integers u and v meaning that node u and node v are connected to each other in undirected fashion.

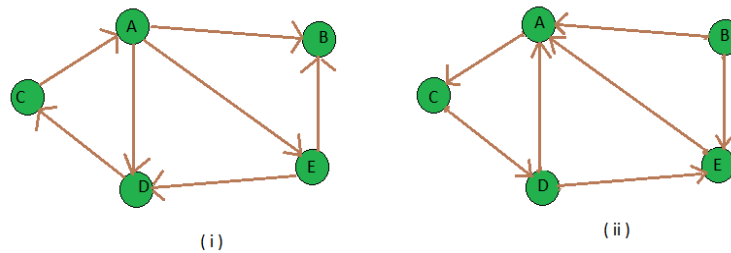
Output: For each input graph print an integer x denoting total number of connected components.

Sample Input	Sample Output
8 5	3
1 2	
2 3	
2 4	
3 5	
6 7	

Constraints: All the input values are well within the integer range.

5.4 Transpose Graph

Transpose of a directed graph G is another directed graph on the same set of vertices with all of the edges reversed compared to the orientation of the corresponding edges in G . That is, if G contains an edge (u, v) then the converse/transpose/reverse of G contains an edge (v, u) and vice versa. Given a graph (represented as adjacency list), we need to find another graph which is the transpose of the given graph.



Input: figure (i) is the input graph.

Output: figure (ii) is the transpose graph of the given graph.

```
0--> 2
1--> 0 4
2--> 3
3--> 0 4
4--> 0
```

Explanation: We traverse the adjacency list and as we find a vertex v in the adjacency list of vertex u which indicates an edge from u to v in main graph, we just add an edge from v to u in the transpose graph i.e. add u in the adjacency list of vertex v of the new graph. Thus traversing lists of all vertices of main graph we can get the transpose graph.

Hints:

```
# find transpose of a graph.
# function to add an edge from vertex source to vertex dest
def addEdge(adj, src, dest):
    adj[src].append(dest)

# function to print adjacency list of a graph
def displayGraph(adj, v):
    ...
    print()

# function to get Transpose of a graph taking adjacency list of given graph and that of Transpose graph
def transposeGraph(adj, transpose, v):
    # traverse the adjacency list of given graph and for each edge (u, v) add
    # an edge (v, u) in the transpose graph's adjacency list
    ...

# Driver Code
v = 5
adj = [[] for i in range(v)]
addEdge(adj, 0, 1)
addEdge(adj, 0, 4)
addEdge(adj, 0, 3)
addEdge(adj, 2, 0)
addEdge(adj, 3, 2)
addEdge(adj, 4, 1)
addEdge(adj, 4, 3)

# Finding transpose of graph represented by adjacency list adj[]
transpose = [[] for i in range(v)]
transposeGraph(adj, transpose, v)

# Displaying adjacency list of transpose graph i.e. b
```

TRY

1. Take input operations as addEdge(A, B), addEdge(A, D), addEdge(A, C), addEdge(C, A), addEdge(A, D), addEdge(C, B), addEdge(B, C) and verify the result.

5.5 Counting Triplets

You are given an undirected, complete graph G that contains N vertices. Each edge is colored in either white or black. You are required to determine the number of triplets (i, j, k) ($1 \leq i < j < k \leq N$) of vertices such that the edges (i, j) , (j, k) , (i, k) are of the same color.

There are M white edges and $(N(N-1)/2) - M$ black edges.

Input:

First line: Two integers – N and M ($3 \leq N \leq 10^5$, $1 \leq M \leq 3 * 10^5$)

$(i+1)^{\text{th}}$ line: Two integers – u_i and v_i ($1 \leq u_i, v_i \leq N$) denoting that the edge (u_i, v_i) is white in color.

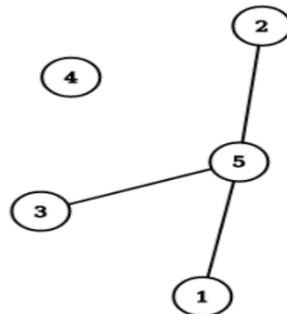
Note: The conditions $(u_i, v_i) \neq (u_j, v_j)$ and $(u_i, v_i) \neq (v_j, u_j)$ are satisfied for all $1 \leq i < j \leq M$.

Output: Print an integer that denotes the number of triples that satisfy the mentioned condition.

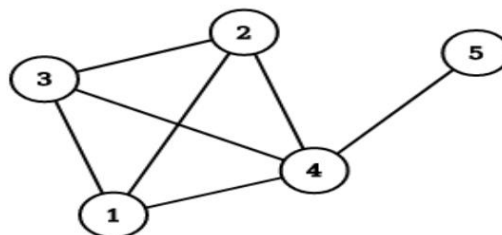
Sample Input	Sample Output
5 3	4
1 5	
2 5	
3 5	

Explanation: The triplets are: $\{(1, 2, 3), (1, 2, 4), (2, 3, 4), (1, 3, 4)\}$

The graph consisting of only white edges:



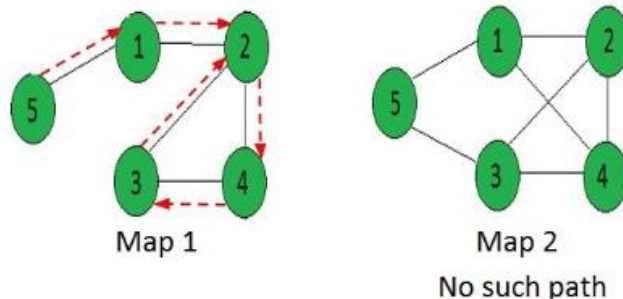
The graph consisting of only black edges:

**6. Graph Routing Algorithms**

6.1 Seven Bridges of Königsberg

There was 7 bridges connecting 4 lands around the city of Königsberg in Prussia. Was there any way to start from any of the land and go through each of the bridges once and only once? Euler first introduced graph theory to solve this problem. He considered each of the lands as a node of a graph and each bridge in between as an edge in between. Now he calculated if there is any Eulerian Path in that graph. If there is an Eulerian path then there is a solution otherwise not.

There are n nodes and m bridges in between these nodes. Print the possible path through each node using each edges (if possible), traveling through each edges only once.



Input: `[[0, 1, 0, 0, 1],
[1, 0, 1, 1, 0],
[0, 1, 0, 1, 0],
[0, 1, 1, 0, 0],
[1, 0, 0, 0, 0]]`

Output: `5 -> 1 -> 2 -> 4 -> 3 -> 2`

Input: `[[0, 1, 0, 1, 1],
[1, 0, 1, 0, 1],
[0, 1, 0, 1, 1],
[1, 1, 1, 0, 0],
[1, 0, 1, 0, 0]]`

Output: `"No Solution"`

Hints:

```
# A Python program to print Eulerian trail in a  
# given Eulerian or Semi-Eulerian Graph  
from collections import defaultdict  
  
class Graph:  
    # Constructor and destructor  
    def __init__(self, V):  
        self.V = V  
        self.adj = defaultdict(list)  
  
    # functions to add and remove edge
```

```

def addEdge(self, u, v):
def rmvEdge(self, u, v):
    ...
# Methods to print Eulerian tour
def printEulerTour(self):
    # Find a vertex with odd degree
    ...
    # Print tour starting from oddv
self.printEulerUtil(u)
print()
def printEulerUtil(self, u):
    # Recur for all the vertices adjacent to this vertex
for v in self.adj[u]:
    # If edge u-v is not removed and it's a valid next edge
    # The function to check if edge u-v can be considered as next edge in Euler Tout
    def isValidNextEdge(self, u, v):
        # The edge u-v is valid in one of the following two cases:

        # 1) If v is the only adjacent vertex of u
        ...

        # 2) If there are multiple adjacents, then u-v is not a bridge
        # Do following steps to check if u-v is a bridge
        # 2.a) count of vertices reachable from u
        ...

        # 2.b) Remove edge (u, v) and after removing
        # the edge, count vertices reachable from u
        ...

        # 2.c) Add the edge back to the graph
        self.addEdge(u, v)

        # 2.d) If count1 is greater, then edge (u, v) is a bridge
        return False if count1 > count2 else True

    # A DFS based function to count reachable vertices from v

    def DFSCount(self, v, visited):
        # Mark the current node as visited
        ...
        # Recur for all the vertices adjacent to this vertex
        ...

    # utility function to form edge between two vertices source and dest
    def makeEdge(src, dest):
        graph.addEdge(src, dest)

# Driver code
# Let us first create and test graphs shown in above figure
g1 = Graph(4)
g1.addEdge(0, 1)
g1.addEdge(0, 2)
g1.addEdge(1, 2)
g1.addEdge(2, 3)
g1.printEulerTour()

g3 = Graph(4)
g3.addEdge(0, 1)

```

```

g3.addEdge(1, 0)
g3.addEdge(0, 2)
g3.addEdge(2, 0)
g3.addEdge(2, 3)
g3.addEdge(3, 1)
g3.printEulerTour()

```

TRY

1. Take input: `[[1, 0, 1, 0, 1], [1, 0, 1, 0, 0], [1, 1, 0, 1, 0], [0, 0, 1, 0, 0], [1, 0, 1, 0, 0]]` and verify the result.
2. Take input: `[[0, 0, 1, 0, 1], [0, 0, 1, 0, 0], [1, 0, 0, 1, 0], [1, 0, 1, 0, 0], [1, 1, 1, 0, 0]]` and verify the result.

6.2 Hamiltonian Cycle

The Hamiltonian cycle of undirected graph $G = \langle V, E \rangle$ is the cycle containing each vertex in V . If graph contains a Hamiltonian cycle, it is called Hamiltonian graph otherwise it is non-Hamiltonian.

Hamiltonian Path in an undirected graph is a path that visits each vertex exactly once. A Hamiltonian cycle (or Hamiltonian circuit) is a Hamiltonian path such that there is an edge (in the graph) from the last vertex to the first vertex of the Hamiltonian path. Consider a graph G , and determine whether a given graph contains Hamiltonian cycle or not. If it contains, then prints the path. Following are the input and output of the required function.

Input: A 2D array graph $[V][V]$ where V is the number of vertices in graph and graph $[V][V]$ is adjacency matrix representation of the graph. A value graph $[i][j]$ is 1 if there is a direct edge from i to j , otherwise graph $[i][j]$ is 0.

Output: An array path $[V]$ that should contain the Hamiltonian Path. Path $[i]$ should represent the i^{th} vertex in the Hamiltonian Path. The code should also return false if there is no Hamiltonian Cycle in the graph.

For example, a Hamiltonian Cycle in the following graph is $\{0, 1, 2, 4, 3, 0\}$.

(0)--(1)--(2)

```

|  /\  |
|  /  \ |
|  /   \ |

```

(3)-----(4)

And the following graph doesn't contain any Hamiltonian Cycle.

(0)--(1)--(2)

```

|  /\  |
|  /  \ |
|  /   \ |

```

(3) (4)

Backtracking Algorithm: Create an empty path array and add vertex 0 to it. Add other vertices, starting from the vertex 1. Before adding a vertex, check for whether it is adjacent to the previously added vertex and not already added. If we find such a vertex, we add the vertex as part of the solution. If we do not find a vertex then we return false.

Hints:

```
# Python program for solution of Hamiltonian cycle problem
```

```

class Graph():
    def __init__(self, vertices):
        self.graph = [[0 for column in range(vertices)]
                      for row in range(vertices)]
        self.V = vertices

    def isSafe(self, v, pos, path):
        # Check if current vertex and last vertex in path are adjacent
        ...

    # A recursive utility function to solve Hamiltonian cycle problem
    def hamCycleUtil(self, path, pos):
        ...

    def hamCycle(self):
        ...

    def printSolution(self, path):
        print ("Solution Exists: Following", "is one Hamiltonian Cycle")
        for vertex in path:
            print (vertex, end = " ")
        print (path[0], "\n")

# Driver Code

""" Let us create the following graph
(0)--(1)--(2)
 |  /\  |
 |  /\  |
 |  /\  |
(3)-----(4) """
g1 = Graph(5)
g1.graph = [ [0, 1, 0, 1, 0], [1, 0, 1, 1, 1],
             [0, 1, 0, 0, 1], [1, 1, 0, 0, 1],
             [0, 1, 1, 1, 0], ]

# Print the solution
g1.hamCycle();

""" Let us create the following graph
(0)--(1)--(2)
 |  /\  |
 |  /\  |
 |  /\  |
(3)   (4) """
g2 = Graph(5)
g2.graph = [ [0, 1, 0, 1, 0], [1, 0, 1, 1, 1],
             [0, 1, 0, 0, 1], [1, 1, 0, 0, 0],
             [0, 1, 1, 0, 0], ]

# Print the solution
g2.hamCycle();

```

TRY

1. Take a graph = [[1, 1, 0, 1, 0], [1, 1, 1, 1, 1], [0, 1, 0, 1, 1], [1, 1, 0, 1, 0], [0, 1, 1, 1, 0],] and verify the results.

6.3 Number of Hamiltonian Cycle

Given an undirected complete graph of N vertices where $N > 2$. The task is to find the number of different Hamiltonian cycle of the graph.

Complete Graph: A graph is said to be complete if each possible vertices is connected through an Edge.

Hamiltonian Cycle: It is a closed walk such that each vertex is visited at most once except the initial vertex. and it is not necessary to visit all the edges.

Formula: $(N - 1)! / 2$

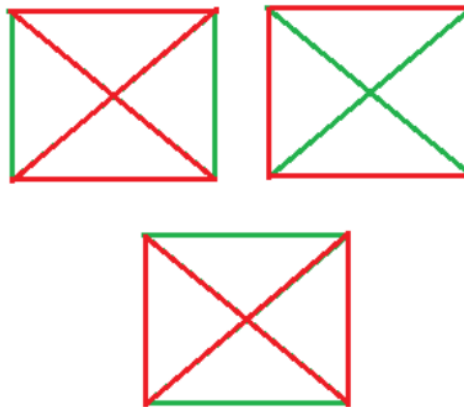
Input: $N = 6$

Output: Hamiltonian cycles = 60

Input: $N = 4$

Output: Hamiltonian cycles = 3

Explanation: Let us take the example of $N = 4$ complete undirected graph, The 3 different Hamiltonian cycle is as shown below:



Hints:

```
# Number of Hamiltonian cycles
import math as mt

# Function that calculates number of Hamiltonian cycle
def Cycles(N):
    ...

# Driver code
N = 5
Number = Cycles(N)
print("Hamiltonian cycles = ", Number)
```

TRY

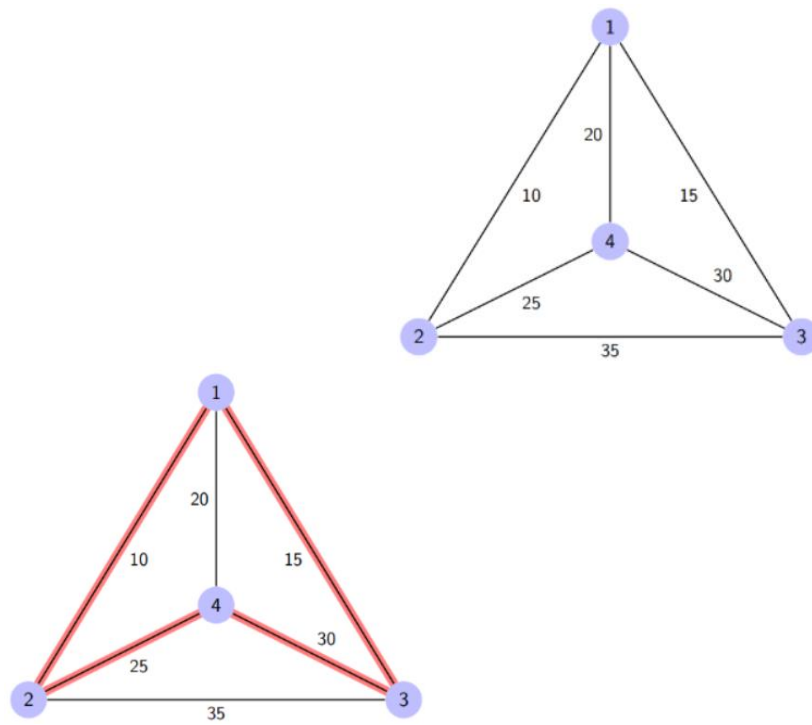
1. Take an input $N=7$ and verify the results.
2. Take an input $N=10$ and verify the results.

7. Shortest Path Algorithms

7.1 Travelling Salesman Problem

Given a set of cities and the distance between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point. The problem statement gives a list of cities along with the distances between each city.

Objective: To start from the origin city, visit other cities only once, and return to the original city again. Our target is to find the shortest possible path to complete the round-trip route.



Here a graph is given where 1, 2, 3, and 4 represent the cities, and the weight associated with every edge represents the distance between those cities. The goal is to find the shortest possible path for the tour that starts from the origin city, traverses the graph while only visiting the other cities or nodes once, and returns to the origin city.

For the above graph, the optimal route is to follow the minimum cost path: 1 – 2 – 4 – 3 – 1. And this shortest route would cost $10 + 25 + 30 + 15 = 80$

Algorithm for Traveling Salesman Problem: We will use the dynamic programming approach to solve the Travelling Salesman Problem (TSP).

- A graph $G=(V, E)$, which is a set of vertices and edges.
- V is the set of vertices.
- E is the set of edges.
- Vertices are connected through edges.
- $\text{Dist}(i,j)$ denotes the non-negative distance between two vertices, i and j .

Let's assume S is the subset of cities and belongs to $\{1, 2, 3, \dots, n\}$ where $1, 2, 3 \dots n$ are the cities and i, j are two cities in that subset. Now $\text{cost}(i, S, j)$ is defined in such a way as the length of the shortest path visiting node in S , which is exactly once having the starting and ending point as i and j respectively.

For example, $\text{cost}(1, \{2, 3, 4\}, 1)$ denotes the length of the shortest path where:

- Starting city is 1
- Cities 2, 3, and 4 are visited only once
- The ending point is 1

The dynamic programming algorithm would be:

- Set $\text{cost}(i, i) = 0$, which means we start and end at i , and the cost is 0.
- When $|S| > 1$, we define $\text{cost}(i, S, 1) = \infty$ where $i \neq 1$. Because initially, we do not know the exact cost to reach city i to city 1 through other cities.
- Now, we need to start at 1 and complete the tour. We need to select the next city in such a way-
- $\text{cost}(i, S, j) = \min \text{cost}(i, S - \{i\}, j) + \text{dist}(i, j)$ where $i \in S$ and $i \neq j$

For the given figure above, the adjacency matrix would be the following:

dist(i, j)	1	2	3	4
1	0	10	15	20
2	10	0	35	25
3	15	35	0	30
4	20	25	30	0

Now $S = \{1, 2, 3, 4\}$. There are four elements. Hence the number of subsets will be 2^4 or 16. Those subsets are-

1) $|S| = \text{Null}$:

$\{\Phi\}$

2) $|S| = 1$:

$\{\{1\}, \{2\}, \{3\}, \{4\}\}$

3) $|S| = 2$:

$\{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}\}$

4) $|S| = 3$:

$\{\{1, 2, 3\}, \{1, 2, 4\}, \{2, 3, 4\}, \{1, 3, 4\}\}$

5) $|S| = 4$:

$\{\{1, 2, 3, 4\}\}$

As we are starting at 1, we could discard the subsets containing city 1. The algorithm calculation steps:

1) $|S| = \Phi$:

$\text{cost}(2, \Phi, 1) = \text{dist}(2, 1) = 10$

$\text{cost}(3, \Phi, 1) = \text{dist}(3, 1) = 15$

$\text{cost}(4, \Phi, 1) = \text{dist}(4, 1) = 20$

2) $|S| = 1$:

$\text{cost}(2, \{3\}, 1) = \text{dist}(2, 3) + \text{cost}(3, \Phi, 1) = 35 + 15 = 50$

$\text{cost}(2, \{4\}, 1) = \text{dist}(2, 4) + \text{cost}(4, \Phi, 1) = 25 + 20 = 45$

$\text{cost}(3, \{2\}, 1) = \text{dist}(3, 2) + \text{cost}(2, \Phi, 1) = 35 + 10 = 45$

$\text{cost}(3, \{4\}, 1) = \text{dist}(3, 4) + \text{cost}(4, \Phi, 1) = 30 + 20 = 50$

$\text{cost}(4, \{2\}, 1) = \text{dist}(4, 2) + \text{cost}(2, \Phi, 1) = 25 + 10 = 35$

$\text{cost}(4, \{3\}, 1) = \text{dist}(4, 3) + \text{cost}(3, \Phi, 1) = 30 + 15 = 45$

3) $|S| = 2$:

$\text{cost}(2, \{3, 4\}, 1) = \min [\text{dist}[2,3] + \text{Cost}(3, \{4\}, 1) = 35 + 50 = 85,$

$\text{dist}[2,4] + \text{Cost}(4, \{3\}, 1) = 25 + 45 = 70] = 70$

$\text{cost}(3, \{2, 4\}, 1) = \min [\text{dist}[3,2] + \text{Cost}(2, \{4\}, 1) = 35 + 45 = 80,$

$\text{dist}[3,4] + \text{Cost}(4, \{2\}, 1) = 30 + 35 = 65] = 65$

$\text{cost}(4, \{2, 3\}, 1) = \min [\text{dist}[4,2] + \text{Cost}(2, \{3\}, 1) = 25 + 50 = 75$

$\text{dist}[4,3] + \text{Cost}(3, \{2\}, 1) = 30 + 45 = 75] = 75$

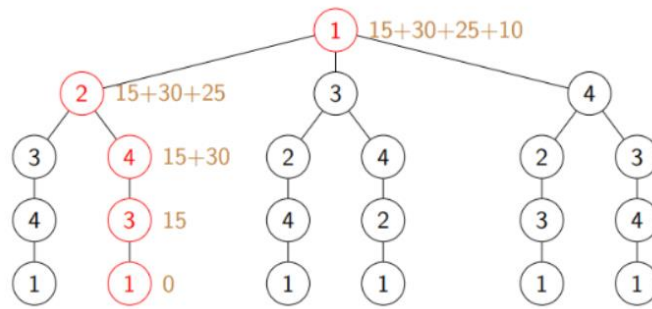
4) $|S| = 3$:

$\text{cost}(1, \{2, 3, 4\}, 1) = \min [\text{dist}[1,2] + \text{Cost}(2, \{3,4\}, 1) = 10 + 70 = 80$

$\text{dist}[1,3] + \text{Cost}(3, \{2,4\}, 1) = 15 + 65 = 80$

$\text{dist}[1,4] + \text{Cost}(4, \{2,3\}, 1) = 20 + 75 = 95] = 80$

So the optimal solution would be 1-2-4-3-1



Hints:

```
from sys import maxsize
from itertools, import permutations
V = 4
def tsp(graph, s):
    ...

# Driver code
graph = [[0, 10, 15, 20], [10, 0, 35, 25], [15, 35, 0, 30], [20, 25, 30, 0]]
s = 0
print(tsp(graph, s))
```

TRY

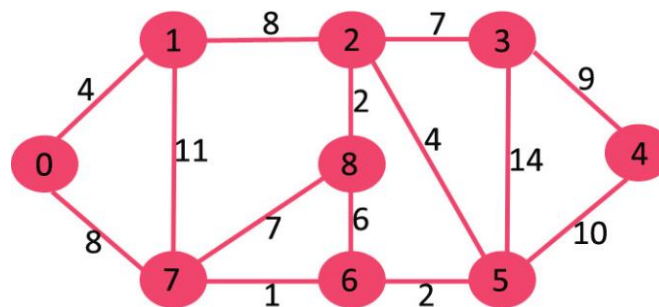
1. Take a below table values and verify the results.

dist(i, j)	1	2	3	4
1	0	40	25	40
2	20	0	35	25
3	25	35	0	60
4	40	25	30	0

7.2 Shortest Paths from Source to all Vertices (Dijkstra's Algorithm)

Given a graph and a source vertex in the graph, find the shortest paths from the source to all vertices in the given graph.

Input: src = 0, the graph is shown below.



Output: 0 4 12 19 21 11 9 8 14

Explanation: The distance from 0 to 1 = 4.

The minimum distance from 0 to 2 = 12. 0->1->2
 The minimum distance from 0 to 3 = 19. 0->1->2->3
 The minimum distance from 0 to 4 = 21. 0->7->6->5->4
 The minimum distance from 0 to 5 = 11. 0->7->6->5
 The minimum distance from 0 to 6 = 9. 0->7->6
 The minimum distance from 0 to 7 = 8. 0->7
 The minimum distance from 0 to 8 = 14. 0->1->2->8

Hints:

```
# Dijkstra's single source shortest path algorithm. The program is for adjacency matrix representation of the graph

# Library for INT_MAX
import sys

class Graph():
    def __init__(self, vertices):
        self.V = vertices
        self.graph = [[0 for column in range(vertices)]
                      for row in range(vertices)]

    def printSolution(self, dist):
        print("Vertex \tDistance from Source")
        for node in range(self.V):
            print(node, "\t", dist[node])

    # A utility function to find the vertex with minimum distance value,
    # from the set of vertices not yet included in shortest path tree
    def minDistance(self, dist, sptSet):
        ...

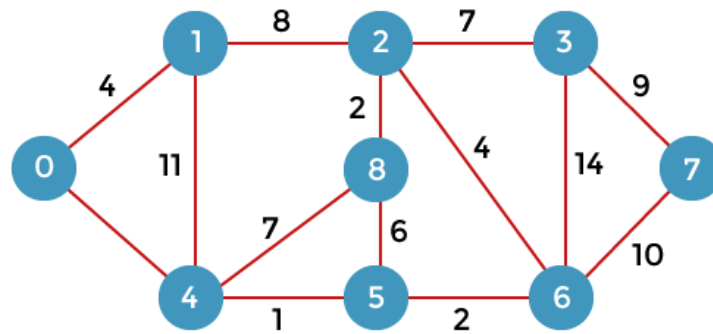
    # Function that implements Dijkstra's single source shortest path
    # algorithm for a graph represented using adjacency matrix representation
    def dijkstra(self, src):
        ...

# Driver's code
g = Graph(9)
g.graph = [[0, 4, 0, 0, 0, 0, 0, 8, 0],
           [4, 0, 8, 0, 0, 0, 0, 11, 0],
           [0, 8, 0, 7, 0, 4, 0, 0, 2],
           [0, 0, 7, 0, 9, 14, 0, 0, 0],
           [0, 0, 0, 9, 0, 10, 0, 0, 0],
           [0, 0, 4, 14, 10, 0, 2, 0, 0],
           [0, 0, 0, 0, 0, 2, 0, 1, 6],
           [8, 11, 0, 0, 0, 0, 1, 0, 7],
           [0, 0, 2, 0, 0, 0, 6, 7, 0]]

g.dijkstra(0)
```

TRY

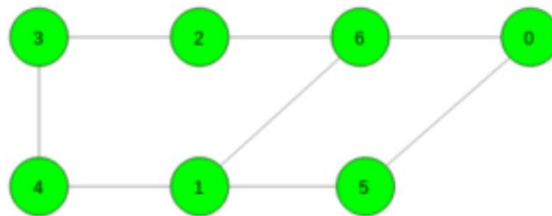
1. Take a below graph and verify the results.



7.3 Shortest Cycle in an Undirected Unweighted Graph

Given an undirected unweighted graph. The task is to find the length of the shortest cycle in the given graph. If no cycle exists print -1.

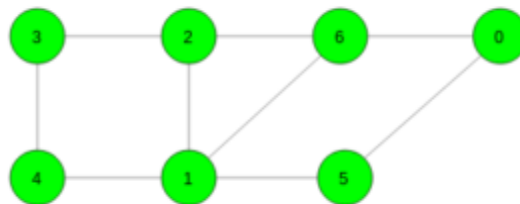
Input: Consider the graph given below



Output: 4

Cycle 6 -> 1 -> 5 -> 0 -> 6

Input: Consider the graph given below



Output: 3

Cycle 6 -> 1 -> 2 -> 6

Hints:

```
from sys import maxsize as INT_MAX
from collections import deque
```

```
N = 100200
```

```
gr = [0] * N
for i in range(N):
    gr[i] = []
```

```
# Function to add edge
```

```

def add_edge(x: int, y: int) -> None:
    global gr
    gr[x].append(y)
    gr[y].append(x)

# Function to find the length of the shortest cycle in the graph
def shortest_cycle(n: int) -> int:

    # To store length of the shortest cycle
    ans = INT_MAX

    # For all vertices
    # write code here
    ...

    # If graph contains no cycle
    if ans == INT_MAX:
        return -1

    # If graph contains cycle
    else:
        return ans

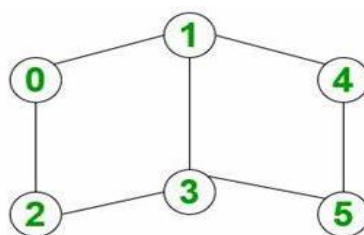
# Driver Code
# Number of vertices
n = 7
# Add edges
add_edge(0, 6)
add_edge(0, 5)
add_edge(5, 1)
add_edge(1, 6)
add_edge(2, 6)
add_edge(2, 3)
add_edge(3, 4)
add_edge(4, 1)

# Function call
print(shortest_cycle(n))

```

TRY

1. Take a below graph and verify the results.



7.4 Count Unique and all Possible Paths in a M x N Matrix

Count unique paths: The problem is to count all unique possible paths from the top left to the bottom right of a M X N matrix with the constraints that from each cell you can either move only to the right or down.

Input: M = 2, N = 2

Output: 2

Explanation: There are two paths

(0, 0) -> (0, 1) -> (1, 1)

(0, 0) -> (1, 0) -> (1, 1)

Input: M = 2, N = 3

Output: 3

Explanation: There are three paths

(0, 0) -> (0, 1) -> (0, 2) -> (1, 2)

(0, 0) -> (0, 1) -> (1, 1) -> (1, 2)

(0, 0) -> (1, 0) -> (1, 1) -> (1, 2)

Count all possible paths: We can recursively move to right and down from the start until we reach the destination and then add up all valid paths to get the answer.

Procedure:

- Create a recursive function with parameters as row and column index
- Call this recursive function for N-1 and M-1
- In the recursive function
 - If N == 1 or M == 1 then return 1
 - else call the recursive function with (N-1, M) and (N, M-1) and return the sum of this
- Print the answer

Hints:

```
# Python program to count all possible paths from top left to bottom right
```

```
# Function to return count of possible paths to reach cell at row number m and column number n from the  
# topmost leftmost cell (cell at 1, 1)
```

```
def numberOfPaths(m, n):
```

```
...
```

```
# Driver program to test above function
```

```
m = 3
```

```
n = 3
```

```
print(numberOfPaths(m, n))
```

TRY

1. Take input : M = 3, N = 2 and verify the results.
2. Take input : M = 2, N = 1 and verify the results.

8. Graph Coloring

8.1 Graph Coloring using Greedy Algorithm

Greedy algorithm is used to assign colors to the vertices of a graph. It doesn't guarantee to use minimum colors, but it guarantees an upper bound on the number of colors. The basic algorithm never uses more than $d+1$ colors where d is the maximum degree of a vertex in the given graph.

Basic Greedy Coloring Algorithm:

1. Color first vertex with first color.
2. Do following for remaining $V-1$ vertices.
 - a) Consider the currently picked vertex and color it with the lowest numbered color that has not been used on any previously colored vertices adjacent to it. If all previously used colors appear on vertices adjacent to v , assign a new color to it.

Hints:

Implement greedy algorithm for graph coloring

```
def addEdge(adj, v, w):  
    adj[v].append(w)  
    # Note: the graph is undirected  
    adj[w].append(v)  
    return adj
```

Assigns colors (starting from 0) to all
vertices and prints the assignment of colors

```
def greedyColoring(adj, V):  
    ...
```

Driver Code

```
g1 = [[] for i in range(5)]  
g1 = addEdge(g1, 0, 1)  
g1 = addEdge(g1, 0, 2)  
g1 = addEdge(g1, 1, 2)  
g1 = addEdge(g1, 1, 3)  
g1 = addEdge(g1, 2, 3)  
g1 = addEdge(g1, 3, 4)  
print("Coloring of graph 1 ")  
greedyColoring(g1, 5)
```

```
g2 = [[] for i in range(5)]  
g2 = addEdge(g2, 0, 1)  
g2 = addEdge(g2, 0, 2)  
g2 = addEdge(g2, 1, 2)  
g2 = addEdge(g2, 1, 4)  
g2 = addEdge(g2, 2, 4)  
g2 = addEdge(g2, 4, 3)  
print("\nColoring of graph 2")  
greedyColoring(g2, 5)
```

Output:

Coloring of graph 1
Vertex 0 ---> Color 0
Vertex 1 ---> Color 1
Vertex 2 ---> Color 2
Vertex 3 ---> Color 0
Vertex 4 ---> Color 1

Coloring of graph 2
Vertex 0 ---> Color 0
Vertex 1 ---> Color 1
Vertex 2 ---> Color 2
Vertex 3 ---> Color 0
Vertex 4 ---> Color 3

8.2 Coloring a Cycle Graph

Given the number of vertices in a Cyclic Graph. The task is to determine the Number of colors required to color the graph so that no two adjacent vertices have the same color.

Approach:

- If the no. of vertices is Even then it is Even Cycle and to color such graph we require 2 colors.
- If the no. of vertices is Odd then it is Odd Cycle and to color such graph we require 3 colors.

Input: Vertices = 3

Output: No. of colors require is: 3

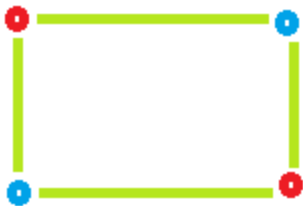
Input: vertices = 4

Output: No. of colors require is: 2

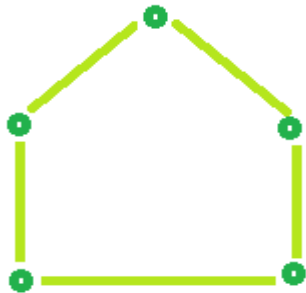
Example 1: Even Cycle: Number of vertices = 4



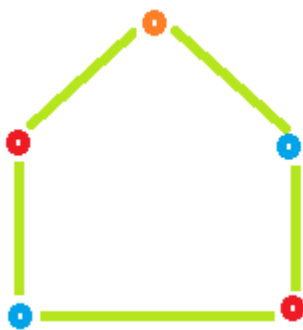
Color required = 2



Example 2: Odd Cycle: Number of vertices = 5



Color required = 3



Hints:

```
# Find the number of colors required to color a cycle graph
```

```
# Function to find Color required.
```

```
def Color(vertices):
```

```
    ...
```

```
# Driver Code
```

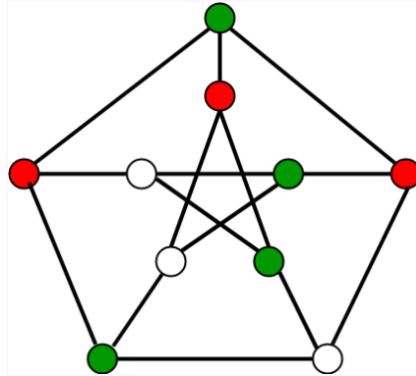
```
vertices = 3
```

```
print ("No. of colors require is:", Color(vertices))
```

8.3 m Coloring Problem

Given an undirected graph and a number m , determine if the graph can be colored with at most m colors such that no two adjacent vertices of the graph are colored with the same color.

Note: Here coloring of a graph means the assignment of colors to all vertices
Following is an example of a graph that can be colored with 3 different colors:



Input: graph = {0, 1, 1, 1},
 {1, 0, 1, 0},
 {1, 1, 0, 1},
 {1, 0, 1, 0}

Output: Solution Exists: Following are the assigned colors: 1 2 3 2

Explanation: By coloring the vertices with following colors, adjacent vertices does not have same colors

Input: graph = {1, 1, 1, 1},
 {1, 1, 1, 1},
 {1, 1, 1, 1},
 {1, 1, 1, 1}

Output: Solution does not exist

Explanation: No solution exists

Generate all possible configurations of colors. Since each node can be colored using any of the m available colors, the total number of color configurations possible is m^V . After generating a configuration of color, check if the adjacent vertices have the same color or not. If the conditions are met, print the combination and break the loop

Follow the given steps to solve the problem:

- Create a recursive function that takes the current index, number of vertices and output color array
- If the current index is equal to number of vertices. Check if the output color configuration is safe, i.e check if the adjacent vertices do not have same color. If the conditions are met, print the configuration and break
- Assign a color to a vertex (1 to m)
- For every assigned color recursively call the function with next index and number of vertices
- If any recursive function returns true break the loop and returns true.

Hints:

```
# Number of vertices in the graph
```

```
# define 4 4
```

```
# check if the colored graph is safe or not
```

```
def isSafe(graph, color):
```

```
    # check for every edge
```

```

for i in range(4):
    for j in range(i + 1, 4):
        if (graph[i][j] and color[j] == color[i]):
            return False
    return True

def graphColoring(graph, m, i, color):
    # write your code here
    ...

# /* A utility function to print solution */

def printSolution(color):
    print("Solution Exists:" " Following are the assigned colors ")
    for i in range(4):
        print(color[i], end=" ")

# Driver code
# /* Create following graph and test whether it is 3 colorable
# (3)---(2)
# | / |
# | / |
# | / |
# (0)---(1)
# */
graph = [
    [0, 1, 1, 1],
    [1, 0, 1, 0],
    [1, 1, 0, 1],
    [1, 0, 1, 0],
]
m = 3 # Number of colors

# Initialize all color values as 0.
# This initialization is needed
# correct functioning of isSafe()
color = [0 for i in range(4)]

# Function call
if (not graphColoring(graph, m, 0, color)):
    print("Solution does not exist")

```

8.4 Edge Coloring of a Graph

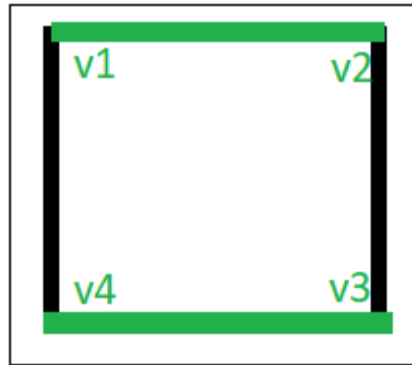
Edge coloring of a graph is an assignment of “colors” to the edges of the graph so that no two adjacent edges have the same color with an optimal number of colors. Two edges are said to be adjacent if they are connected to the same vertex. There is no known polynomial time algorithm for edge-coloring every graph with an optimal number of colors.

Input: $u_1 = 1, v_1 = 4$
 $u_2 = 1, v_2 = 2$
 $u_3 = 2, v_3 = 3$
 $u_4 = 3, v_4 = 4$

Output: Edge 1 is of color 1
 Edge 2 is of color 2
 Edge 3 is of color 1
 Edge 4 is of color 2

The above input shows the pair of vertices (u_i, v_i) who have an edge between them. The output shows the color

assigned to the respective edges.



Edge colorings are one of several different types of graph coloring problems. The above figure of a Graph shows an edge coloring of a graph by the colors green and black, in which no adjacent edge have the same color.

Algorithm:

1. Use BFS traversal to start traversing the graph.
2. Pick any vertex and give different colors to all of the edges connected to it, and mark those edges as colored.
3. Traverse one of its edges.
4. Repeat step 2 with a new vertex until all edges are colored.

Hints:

Edge Coloring

```
from queue import Queue
```

function to determine the edge colors

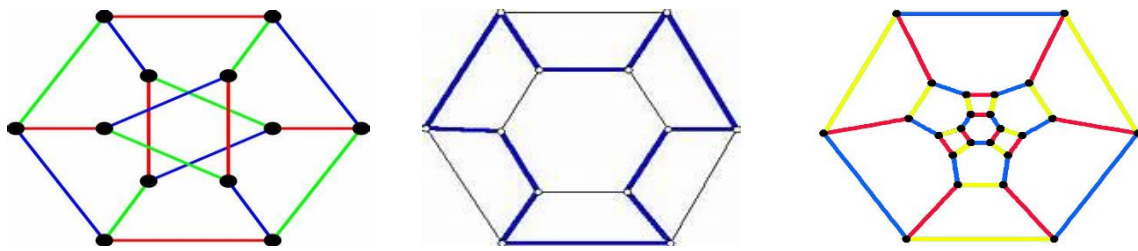
```
def colorEdges(ptr, gra, edgeColors, isVisited):  
    # Write your code here  
    ...
```

Driver Function

```
empty=set()  
# declaring vector of vector of pairs, to define Graph  
gra=[]  
edgeColors=[]  
isVisited=[False]*100000  
  
ver = 4  
edge = 4  
gra=[[] for _ in range(ver)]  
edgeColors=[-1]*edge  
gra[0].append((1, 0))  
gra[1].append((0, 0))  
gra[1].append((2, 1))  
gra[2].append((1, 1))  
gra[2].append((3, 2))  
gra[3].append((2, 2))  
gra[0].append((3, 3))  
gra[3].append((0, 3))  
colorEdges(0, gra, edgeColors, isVisited)  
  
# printing all the edge colors  
for i in range(edge):  
    print("Edge { } is of color { }".format(i + 1, edgeColors[i] + 1))
```

TRY

1. Write a program to implement graph coloring and edge coloring by consider the below graph and verify the results.



9. Graph Traversal

9.1 Breadth First Search

The **Breadth First Search (BFS)** algorithm is used to search a graph data structure for a node that meets a set of criteria. It starts at the root of the graph and visits all nodes at the current depth level before moving on to the nodes at the next depth level.

For a given graph G, print BFS traversal from a given source vertex.

Hints:

BFS traversal from a given source vertex.

```
from collections import defaultdict
```

This class represents a directed graph using adjacency list representation
class Graph:

Constructor

```
def __init__(self):
```

Default dictionary to store graph

```
self.graph = defaultdict(list)
```

Function to add an edge to graph

```
def addEdge(self, u, v):
```

```
self.graph[u].append(v)
```

Function to print a BFS of graph

```
def BFS(self, s):
```

```
# Write your code here
```

```
...
```

Create a graph given in the above diagram

```
g = Graph()
```

```
g.addEdge(0, 1)
```

```
g.addEdge(0, 2)
```

```
g.addEdge(1, 2)
```

```
g.addEdge(2, 0)
```

```
g.addEdge(2, 3)
```

```
g.addEdge(3, 3)
```

```
print("Following is Breadth First Traversal" " (starting from vertex 2)")
```

```
g.BFS(2)
```

Output: Following is Breadth First Traversal (starting from vertex 2)

2 0 3 1

9.2 Depth First Search

Depth First Traversal (or DFS) for a graph is similar to Depth First Traversal of a tree. The only catch here is, that, unlike trees, graphs may contain cycles (a node may be visited twice). To avoid processing a node more than once, use a boolean visited array. A graph can have more than one DFS traversal.

For a given graph G, print DFS traversal from a given source vertex.

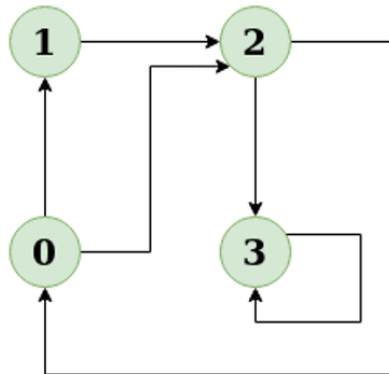
Input: n = 4, e = 6

0 -> 1, 0 -> 2, 1 -> 2, 2 -> 0, 2 -> 3, 3 -> 3

Output: DFS from vertex 1: 1 2 0 3

Explanation:

DFS Diagram:



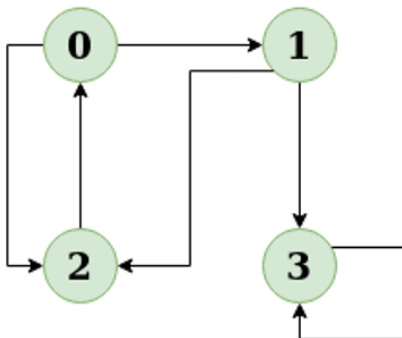
Input: n = 4, e = 6

2 -> 0, 0 -> 2, 1 -> 2, 0 -> 1, 3 -> 3, 1 -> 3

Output: DFS from vertex 2: 2 0 1 3

Explanation:

DFS Diagram:



Hints:

```
# DFS traversal from a given graph
from collections import defaultdict
```

```
# This class represents a directed graph using adjacency list representation
class Graph:
```

```
    # Constructor
```

```
    def __init__(self):
```

```
        # Default dictionary to store graph
```

```
        self.graph = defaultdict(list)
```

```
    # Function to add an edge to graph
```

```

def addEdge(self, u, v):
    self.graph[u].append(v)

# A function used by DFS
def DFSUtil(self, v, visited):
    ...

# The function to do DFS traversal. It uses recursive DFSUtil()

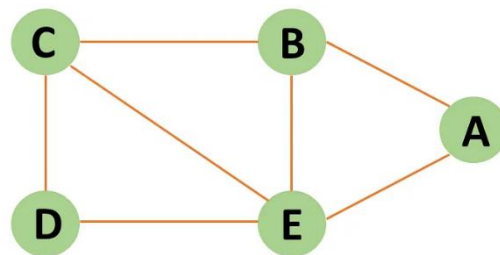
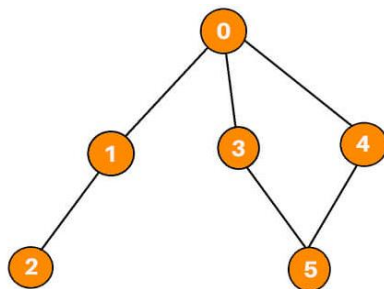
def DFS(self, v):
    # Write your code here
    ...

# Driver's code
g = Graph()
g.addEdge(0, 1)
g.addEdge(0, 2)
g.addEdge(1, 2)
g.addEdge(2, 0)
g.addEdge(2, 3)
g.addEdge(3, 3)
print("Following is Depth First Traversal (starting from vertex 2)")
# Function call
g.DFS(2)

```

TRY

1. Write a program to implement breadth first search and depth first search by consider the below graph and verify the results.



10. Minimum Spanning Tree (MST)

10.1 Kruskal's Algorithm

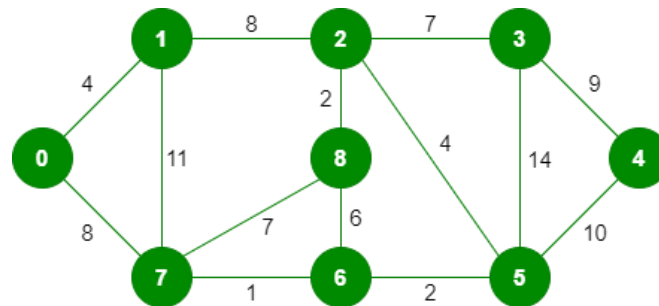
In Kruskal's algorithm, sort all edges of the given graph in increasing order. Then it keeps on adding new edges and nodes in the MST if the newly added edge does not form a cycle. It picks the minimum weighted edge at first and the maximum weighted edge at last.

MST using Kruskal's algorithm:

1. Sort all the edges in non-decreasing order of their weight.
2. Pick the smallest edge. Check if it forms a cycle with the spanning tree formed so far. If the cycle is not formed, include this edge. Else, discard it.
3. Repeat step#2 until there are $(V-1)$ edges in the spanning tree.

Kruskal's algorithm to find the minimum cost spanning tree uses the greedy approach. The Greedy Choice is to pick the smallest weight edge that does not cause a cycle in the MST constructed so far.

Input: For the given graph G find the minimum cost spanning tree.



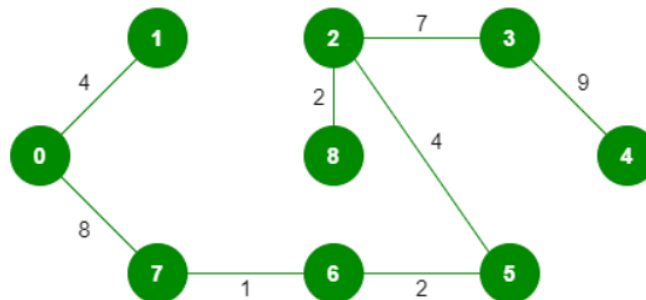
The graph contains 9 vertices and 14 edges. So, the minimum spanning tree formed will be having $(9 - 1) = 8$ edges.

After sorting:

Weight	Source	Destination
1	7	6
2	8	2
2	6	5
4	0	1
4	2	5
6	8	6
7	2	3
7	7	8
8	0	7
8	1	2
9	3	4
10	5	4
11	1	7
14	3	5

Now pick all edges one by one from the sorted list of edges.

Output:



Hints:

```
# Kruskal's algorithm to find minimum Spanning Tree of a given connected,
# undirected and weighted graph
```

```
# Class to represent a graph
```

```
class Graph:
```

```
    def __init__(self, vertices):
```

```
        self.V = vertices
```

```
        self.graph = []
```

```
# Function to add an edge to graph
```

```
def addEdge(self, u, v, w):
```



```

        self.graph.append([u, v, w])

    def find(self, parent, i):
        ...

    def union(self, parent, rank, x, y):
        ...

    def KruskalMST(self):
        # write your code here
        ...

# Driver code
g = Graph(4)
g.addEdge(0, 1, 10)
g.addEdge(0, 2, 6)
g.addEdge(0, 3, 5)
g.addEdge(1, 3, 15)
g.addEdge(2, 3, 4)

# Function call
g.KruskalMST()

```

Output: Following are the edges in the constructed MST

2 -- 3 == 4

0 -- 3 == 5

0 -- 1 == 10

Minimum Cost Spanning Tree: 19

10.2 Prim's Algorithm

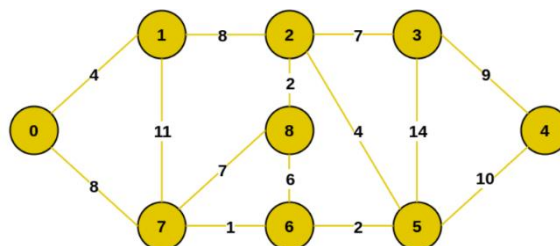
The Prim's algorithm starts with an empty spanning tree. The idea is to maintain two sets of vertices. The first set contains the vertices already included in the MST, and the other set contains the vertices not yet included. At every step, it considers all the edges that connect the two sets and picks the minimum weight edge from these edges. After picking the edge, it moves the other endpoint of the edge to the set containing MST.

Prim's Algorithm:

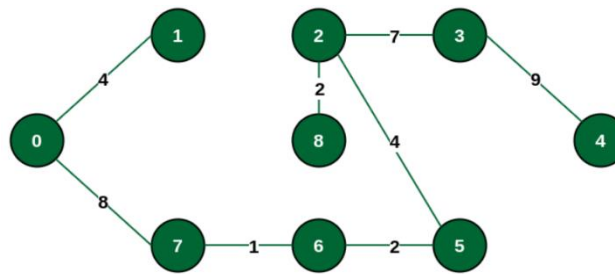
The working of Prim's algorithm can be described by using the following steps:

1. Determine an arbitrary vertex as the starting vertex of the MST.
2. Follow steps 3 to 5 till there are vertices that are not included in the MST (known as fringe vertex).
3. Find edges connecting any tree vertex with the fringe vertices.
4. Find the minimum among these edges.
5. Add the chosen edge to the MST if it does not form any cycle.
6. Return the MST and exit

Input: For the given graph G find the minimum cost spanning tree.



Output: The final structure of the MST is as follows and the weight of the edges of the MST is $(4 + 8 + 1 + 2 + 4 + 2 + 7 + 9) = 37$.



Hints:

Prim's Minimum Spanning Tree (MST) algorithm.
The program is for adjacency matrix representation of the graph

Library for INT_MAX
import sys

```

class Graph():
    def __init__(self, vertices):
        self.V = vertices
        self.graph = [[0 for column in range(vertices)]
                      for row in range(vertices)]

    # A utility function to print
    # the constructed MST stored in parent[]
    def printMST(self, parent):
        print("Edge \tWeight")
        for i in range(1, self.V):
            print(parent[i], "-", i, "\t", self.graph[i][parent[i]])

    # A utility function to find the vertex with
    # minimum distance value, from the set of vertices
    # not yet included in shortest path tree
    def minKey(self, key, mstSet):

        ...

    def primMST(self):
        ...

```

```

# Driver's code
g = Graph(5)
g.graph = [[0, 2, 0, 6, 0],
           [2, 0, 3, 8, 5],
           [0, 3, 0, 0, 7],
           [6, 8, 0, 0, 9],
           [0, 5, 7, 9, 0]]

g.primMST()

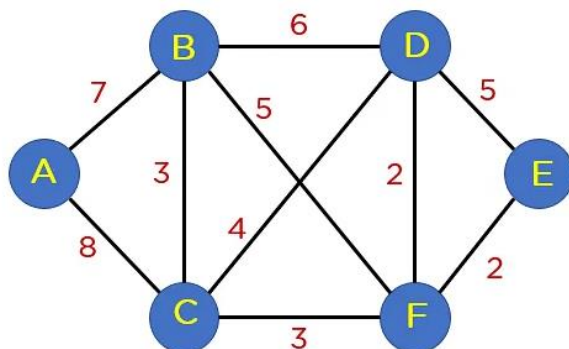
```

Output:

Edge	Weight
0 - 1	2
1 - 2	3
0 - 3	6

TRY

1. Write a program to implement kruskal's algorithm and prim's algorithm by consider the below graph and verify the results.



11. Roots of Equations

11.1 Bisection Method

The Bisection method is also called the interval halving method, the binary search method or the dichotomy method. This method is used to find root of an equation in a given interval that is value of 'x' for which $f(x) = 0$. The method is based on **The Intermediate Value Theorem** which states that if $f(x)$ is a continuous function and there are two real numbers a and b such that $f(a) * f(b) < 0$ and $f(b) < 0$, then it is guaranteed that it has at least one root between them.

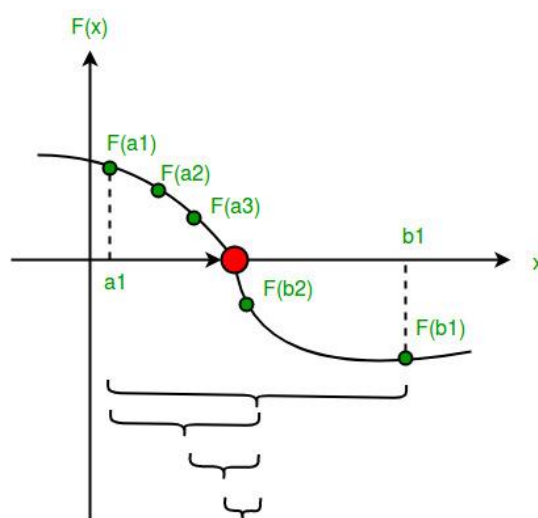
Assumptions:

1. $f(x)$ is a continuous function in interval $[a, b]$
2. $f(a) * f(b) < 0$

Steps:

1. Find middle point $c = (a + b)/2$.
2. If $f(c) == 0$, then c is the root of the solution.
3. Else $f(c) != 0$
 - If value $f(a)*f(c) < 0$ then root lies between a and c . So we recur for a and c
 - Else If $f(b)*f(c) < 0$ then root lies between b and c . So we recur b and c .
 - Else given function doesn't follow one of assumptions.

Since root may be a floating point number, we repeat above steps while difference between a and b is greater than and equal to a value? (A very small value).



Hints:

```

# An example function whose solution is determined using Bisection Method.
# The function is  $x^3 - x^2 + 2$ 
def func(x):
    return  $x*x*x - x*x + 2$ 

# Prints root of func(x) with error of EPSILON
def bisection(a,b):
    # Write your code here
    ...

# Driver code
# Initial values assumed
a = -200
b = 300
bisection (a, b)

```

Output: The value of root is : -1.0025

TRY

1. Take an input function $x^2 - x^3 + 2$ and verify the results.
2. Take an input function $x^3 - x^3 + 4$ and verify the results.

11.2 Method of False Position

Given a function $f(x)$ on floating number x and two numbers 'a' and 'b' such that $f(a)*f(b) < 0$ and $f(x)$ is continuous in $[a, b]$. Here $f(x)$ represents algebraic or transcendental equation. Find root of function in interval $[a, b]$ (Or find a value of x such that $f(x)$ is 0).

Input: A function of x , for example $x^3 - x^2 + 2$.
 And two values: $a = -200$ and $b = 300$ such that
 $f(a)*f(b) < 0$, i.e., $f(a)$ and $f(b)$ have opposite signs.

Output: The value of root is : -1.00
 OR any other value close to root.

Hints:

```

MAX_ITER = 1000000

# An example function whose solution is determined using Regular Falsi Method.
# The function is  $x^3 - x^2 + 2$ 
def func( x ):
    return (x * x * x - x * x + 2)

# Prints root of func(x) in interval [a, b]
def regulaFalsi( a , b):
    # Write your code here
    ...

# Driver code to test above function
# Initial values assumed
a = -200
b = 300
regulaFalsi(a, b)

```

TRY

1. Take an input function $x^2 - x^3 + 2$ and verify the results.
2. Take an input function $x^3 - x^3 + 4$ and verify the results.

11.3 Newton Raphson Method

Given a function $f(x)$ on floating number x and an initial guess for root, find root of function in interval. Here $f(x)$ represents algebraic or transcendental equation.

Input: A function of x (for example $x^3 - x^2 + 2$), derivative function of x ($3x^2 - 2x$ for above example) and an initial guess $x_0 = -20$

Output: The value of root is: -1.00 or any other value close to root.

Algorithm:

Input: initial x , $\text{func}(x)$, $\text{derivFunc}(x)$

Output: Root of $\text{Func}()$

1. Compute values of $\text{func}(x)$ and $\text{derivFunc}(x)$ for given initial x
2. Compute h : $h = \text{func}(x) / \text{derivFunc}(x)$
3. While h is greater than allowed error ϵ
 - $h = \text{func}(x) / \text{derivFunc}(x)$
 - $x = x - h$

Hints:

```
# Implementation of Newton Raphson Method for solving equations
# An example function whose solution is determined using Bisection Method.
# The function is  $x^3 - x^2 + 2$ 
def func( x ):
    return x * x * x - x * x + 2

# Derivative of the above function which is  $3*x^2 - 2*x$ 
def derivFunc( x ):
    return 3 * x * x - 2 * x

# Function to find the root
def newtonRaphson( x ):
    # Write your code here
    ...

# Driver program
x0 = -20
newtonRaphson(x0)
```

TRY

1. Take an input function $x^2 - x^3 + 2$ and verify the results.
2. Take an input function $x^3 - x^3 + 4$ and verify the results.

11.4 Secant Method

The secant method is used to find the root of an equation $f(x) = 0$. It is started from two distinct estimates x_1 and x_2 for the root. It is an iterative procedure involving linear interpolation to a root. The iteration stops if the difference between two intermediate values is less than the convergence factor.

Input: Equation = $x^3 + x - 1$
 $x_1 = 0$, $x_2 = 1$, $E = 0.0001$

Output: Root of the given equation = 0.682326
No. of iteration=5

Algorithm

Initialize: x_1 , x_2 , E , n // E = convergence indicator
calculate $f(x_1)$, $f(x_2)$

```
if( $f(x_1) * f(x_2) = E$ ); //repeat the loop until the convergence
    print 'x0' //value of the root
    print 'n' //number of iteration
}
else
    print "cannot found a root in the given interval"
```

Hints:

```

# Find root of an equations using secant method
# Function takes value of x and returns f(x)
def f(x):
    # we are taking equation as x^3+x-1
    f = pow(x, 3) + x - 1;
    return f;

def secant(x1, x2, E):
    # Write your code here
    ...

# Driver code
# initializing the values
x1 = 0;
x2 = 1;
E = 0.0001;
secant(x1, x2, E);

```

TRY

1. Take an input function $x^2 - x + 2$ and verify the results.
2. Take an input function $x^3 - x^2 + 4$ and verify the results.

11.5 Muller Method

Given a function $f(x)$ on floating number x and three initial distinct guesses for root of the function, find the root of function. Here, $f(x)$ can be an algebraic or transcendental function.

Input: A function $f(x) = x^3 + 2x^2 + 10x - 20$ and three initial guesses - 0, 1 and 2 .

Output: The value of the root is 1.3688 or any other value within permissible deviation from the root.

Input: A function $f(x) = x^3 - 5x + 2$ and three initial guesses - 0, 1 and 2.

Output: The value of the root is 0.4021 or any other value within permissible deviation from the root.

Hints:

```

# Find root of a function, f(x)
import math;

MAX_ITERATIONS = 10000;

# Function to calculate f(x)
def f(x):
    # Taking f(x) = x^3 + 2x^2 + 10x - 20
    return (1 * pow(x, 3) + 2 * x * x +
            10 * x - 20);

def Muller(a, b, c):
    # Write your code here
    ...

# Driver Code
a = 0;
b = 1;
c = 2;
Muller(a, b, c);

```

TRY

1. Take an input function $x^2 - x^3 + 2$ and verify the results.
2. Take an input function $x^3 - x^3 + 4$ and verify the results.

12. Numerical Integration

12.1 Trapezoidal Rule for Approximate Value of Definite Integral

Trapezoidal rule is used to find the approximation of a definite integral. The basic idea in Trapezoidal rule is to assume the region under the graph of the given function to be a trapezoid and calculate its area.

$$\int_a^b f(x) dx \approx (b - a) \left[\frac{f(a) + f(b)}{2} \right]$$

Hints:

```
# Implement Trapezoidal rule

# A sample function whose definite integral's approximate value is
# computed using Trapezoidal rule
def y( x ):

    # Declaring the function
    # f(x) = 1/(1+x*x)
    return (1 / (1 + x * x))

# Function to evaluate the value of integral
def trapezoidal (a, b, n):
    # Write your code here
    ...

# Driver code
# Range of definite integral
x0 = 0
xn = 1

# Number of grids. Higher value means more accuracy
n = 6
print ("Value of integral is ",
      "%.4f"%trapezoidal(x0, xn, n))
```

12.2 Simpson's 1/3 Rule

Simpson's 1/3 rule is a method for numerical approximation of definite integrals. Specifically, it is the following approximation:

$$\int_a^b f(x) dx \approx \frac{(b - a)}{6} \left(f(a) + 4f\left(\frac{a + b}{2}\right) + f(b) \right)$$

Procedure:

In order to integrate any function $f(x)$ in the interval (a, b) , follow the steps given below:

1. Select a value for n , which is the number of parts the interval is divided into.
2. Calculate the width, $h = (b-a)/n$
3. Calculate the values of x_0 to x_n as $x_0 = a$, $x_1 = x_0 + h$, ..., $x_{n-1} = x_{n-2} + h$, $x_n = b$. Consider $y = f(x)$. Now find the values of y (y_0 to y_n) for the corresponding x (x_0 to x_n) values.

4. Substitute all the above found values in the Simpson's Rule Formula to calculate the integral value.

Approximate value of the integral can be given by **Simpson's Rule**:

$$\int_a^b f(x)dx \approx \frac{h}{3} \left(f_0 + f_n + 4 * \sum_{i=1,3,5}^{n-1} f_i + 2 * \sum_{i=2,4,6}^{n-2} f_i \right)$$

Input: Evaluate $\log x$ dx within limit 4 to 5.2.

First we will divide interval into six equal parts as number of interval should be even.

x : 4 4.2 4.4 4.6 4.8 5.0 5.2

\log_x : 1.38 1.43 1.48 1.52 1.56 1.60 1.64

Output: Now we can calculate approximate value of integral using above formula:

$$\begin{aligned} &= h/3[(1.38 + 1.64) + 4 * (1.43 + 1.52 + 1.60) + 2 * (1.48 + 1.56)] \\ &= 1.84 \end{aligned}$$

Hence the approximation of above integral is

1.827 using Simpson's 1/3 rule.

Hints:

```
# Simpson's 1 / 3 rule
import math

# Function to calculate f(x)
def func(x):
    return math.log(x)

# Function for approximate integral
def simpsons_(ll, ul, n):
    # Write your code here
    ...

# Driver code
lower_limit = 4    # Lower limit
upper_limit = 5.2    # Upper limit
n = 6    # Number of interval
print("%.6f"% simpsons_(lower_limit, upper_limit, n))
```

12.3 Simpson's 3/8 Rule

The Simpson's 3/8 rule was developed by Thomas Simpson. This method is used for performing numerical integrations. This method is generally used for numerical approximation of definite integrals. Here, parabolas are used to approximate each part of curve.

Input: lower_limit = 1, upper_limit = 10, interval_limit = 10

Output: integration_result = 0.687927

Input: lower_limit = 1, upper_limit = 5, interval_limit = 3

Output: integration_result = 0.605835

Hints:

```
# Implement Simpson's 3/8 rule

# Given function to be integrated
def func(x):
    return (float(1) / ( 1 + x * x ))
```



```

# Function to perform calculations
def calculate(lower_limit, upper_limit, interval_limit ):
    # Write your code here
    ...

# driver function
interval_limit = 10
lower_limit = 1
upper_limit = 10

integral_res = calculate(lower_limit, upper_limit, interval_limit)

# rounding the final answer to 6 decimal places
print (round(integral_res, 6))

```

13. Ordinary Differential Equations

13.1 The Euler Method

Given a differential equation $dy/dx = f(x, y)$ with initial condition $y(x_0) = y_0$. Find its approximate solution using Euler method.

Euler Method:

In mathematics and computational science, the Euler method (also called forward Euler method) is a first-order numerical procedure for solving ordinary differential equations (ODEs) with a given initial value.

Consider a differential equation $dy/dx = f(x, y)$ with initial condition $y(x_0) = y_0$ then a successive approximation of this equation can be given by:

$$y(n+1) = y(n) + h * f(x(n), y(n))$$

where $h = (x(n) - x(0)) / n$, h indicates step size. Choosing smaller values of h leads to more accurate results and more computation time.

Example:

Consider below differential equation $dy/dx = (x + y + xy)$ with initial condition $y(0) = 1$ and step size $h = 0.025$. Find $y(0.1)$.

Solution:

$$f(x, y) = (x + y + xy)$$

$$x_0 = 0, y_0 = 1, h = 0.025$$

Now we can calculate y_1 using Euler formula

$$y_1 = y_0 + h * f(x_0, y_0)$$

$$y_1 = 1 + 0.025 * (0 + 1 + 0 * 1)$$

$$y_1 = 1.025$$

$$y(0.025) = 1.025.$$

Similarly we can calculate $y(0.050)$, $y(0.075)$, ..., $y(0.1)$.

$$y(0.1) = 1.11167$$

Hints:

Find approximation of an ordinary differential equation using Euler method.

Consider a differential equation

$$\# dy / dx = (x + y + xy)$$

def func(x, y):

 return (x + y + x * y)

Function for Euler formula

def euler(x0, y, h, x):

 # write code here

...

```
# Driver Code
# Initial Values
x0 = 0
y0 = 1
h = 0.025

# Value of x at which we need approximation
x = 0.1
euler(x0, y0, h, x)
```

13.2 Runge-Kutta Second Order Method

Given the following inputs:

1. An ordinary differential equation that defines the value of $\frac{dy}{dx}$ in the form \mathbf{x} and \mathbf{y} .

$$\frac{dy}{dx} = f(x, y)$$
2. Initial value of y , i.e., $\mathbf{y(0)}$.

$$y(0) = y_0$$

The task is to find the value of unknown function y at a given point x , i.e. $\mathbf{y(x)}$.

Input: $x_0 = 0, y_0 = 1, x = 2, h = 0.2$

Output: $y(x) = 0.645590$

Input: $x_0 = 2, y_0 = 1, x = 4, h = 0.4;$

Output: $y(x) = 4.122991$

Approach:

The Runge-Kutta method finds an approximate value of y for a given x . Only first-order ordinary differential equations can be solved by using the Runge-Kutta 2nd-order method.

Below is the formula used to compute the next value y_{n+1} from the previous value y_n . Therefore:

y_{n+1} = value of y at $(x = n + 1)$

y_n = value of y at $(x = n)$

where $0 \leq n \leq (x - x_0)/h$, h is step height

$x_{n+1} = x_0 + h$

The essential formula to compute the value of $y(n+1)$:

$K_1 = h * f(x, y)$

$K_2 = h * f(x/2, y/2)$ or $K_1/2$

$y_{n+1} = y_n + K^2 + (h^3)$

The formula basically computes the next value y_{n+1} using current y_n plus the weighted average of two increments:

- **K1** is the increment based on the slope at the beginning of the interval, using y .
- **K2** is the increment based on the slope at the midpoint of the interval, using $(y + h*K_1/2)$.

Hints:

```
# Implement Runge-Kutta method
```

```
# A sample differential equation
```

```
# "dy/dx = (x - y)/2"
```

```
def dydx(x, y):
    return (x + y - 2)
```

```
# Finds value of y for a given x using step size h and initial value y0 at x0.
```

```
def rungeKutta(x0, y0, x, h):
```

```
    # write code here
```

```
...
# Driver Code
x0 = 0
y = 1
x = 2
h = 0.2
print("y(x) =", rungeKutta(x0, y, x, h))
```

14. Final Notes

The only way to learn programming is program, program and program on challenging problems. The problems in this tutorial are certainly NOT challenging. There are tens of thousands of challenging problems available – used in training for various programming contests (such as International Collegiate Programming Contest (ICPC), International Olympiad in Informatics (IOI)). Check out these sites:

- The ACM - ICPC International collegiate programming contest (<https://icpc.global/>)
- The Topcoder Open (TCO) annual programming and design contest (<https://www.topcoder.com/>)
- Universidad de Valladolid's online judge (<https://uva.onlinejudge.org/>).
- Peking University's online judge (<http://poj.org/>).
- USA Computing Olympiad (USACO) Training Program @ <http://train.usaco.org/usacogate>.
- Google's coding competitions (<https://codingcompetitions.withgoogle.com/codejam>,
<https://codingcompetitions.withgoogle.com/hashcode>)
- The ICFP programming contest (<https://www.icfpconference.org/>)
- BME International 24-hours programming contest (<https://www.challenge24.org/>)
- The International Obfuscated C Code Contest (<https://www0.us.ioccc.org/main.html>)
- Internet Problem Solving Contest (<https://ipsc.ksp.sk/>)
- Microsoft Imagine Cup (<https://imaginecup.microsoft.com/en-us>)
- Hewlett Packard Enterprise (HPE) Codewars (<https://hpecodewars.org/>)
- OpenChallenge (<https://www.openchallenge.org/>)

Coding Contests Scores

Students must solve problems and attain scores in the following coding contests:

	Name of the contest	Minimum number of problems to solve	Required score
•	CodeChef	20	200
•	Leetcode	20	200
•	GeeksforGeeks	20	200
•	SPOJ	5	50
•	InterviewBit	10	1000
•	Hackerrank	25	250
•	Codeforces	10	100
•	BuildIT	50	500
	Total score need to obtain		2500

Student must have any one of the following certification:

1. HackerRank - Problem Solving Skills Certification (Basic and Intermediate)
2. GeeksforGeeks – Data Structures and Algorithms Certification
3. CodeChef - Learn Python Certification
4. Interviewbit – DSA pro / Python pro
5. NPTEL – Programming, Data Structures and Algorithms
6. NPTEL – The Joy of Computing using Python

V. TEXT BOOKS:

1. Eric Matthes, “Python Crash Course: A Hands-On, Project-based Introduction to Programming”, No Starch Press, 3rd Edition, 2023.
2. John M Zelle, “Python Programming: An Introduction to Computer Science”, Ingram short title, 3rd Edition, 2016.

VI. REFERENCE BOOKS:

1. Yashavant Kanetkar, Aditya Kanetkar, “Let Us Python”, BPB Publications, 2nd Edition, 2019.
2. Martin C. Brown, “Python: The Complete Reference”, Mc. Graw Hill, Indian Edition, 2018.

3. Paul Barry, “Head First Python: A Brain-Friendly Guide”, O’Reilly, 2nd Edition, 2016
4. Taneja Sheetal, Kumar Naveen, “Python Programming – A Modular Approach”, Pearson, 1st Edition, 2017.
5. R Nageswar Rao, “Core Python Programming”, Dreamtech Press, 2018.

VII. ELECTRONICS RESOURCES

1. <https://realPython.com/Python3-object-oriented-programming/>
2. <https://Python.swaroopch.com/oop.html>
3. https://Python-textbok.readthedocs.io/en/1.0/Object_Oriented_Programming.html
4. <https://www.programiz.com/Python-programming/>
5. <https://www.geeksforgeeks.org/python-programming-language/>

VIII. MATERIALS ONLINE

1. Course template
2. Lab Manual

COURSE CONTENT

WEB AND MOBILE APPLICATIONS DEVELOPMENT								
II Semester: AE / ME / CE / ECE / EEE								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
ACSD07	Skill	L	T	P	C	CIA	SEE	Total
		0	0	2	1	40	60	100
Contact Classes: Nil	Tutorial Classes: Nil	Practical Classes: 45			Total Classes: 45			
Prerequisite: There is no prerequisite required to this course								

I. COURSE OVERVIEW:

This course serves as foundation course on Web and Mobile applications development. It covers fundamental concepts to build modern web applications, the basics of HTML5 and CSS3 for Web application development. This environment will gain practical skills in mobile app development including user interface design, programming, and deployment.

II. COURSES OBJECTIVES:

The students will try to learn

- I. The characteristics, systematic methods, model for developing web applications.
- II. The concepts of client side programming with Bootstrap ,JavaScript, Ajax , Design user interfaces that follow best practices for usability and user experience
- III. The mobile application development for different platforms using appropriate tools and frameworks.
- IV. The user interface design with best practices for usability and user experience.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO1 Develop web pages using the HTML and CSS features with different layouts as per need of applications
- CO2 Use the JavaScript to develop the dynamic web pages.
- CO3 Use server-side scripting with PHP to generate the web pages dynamically using the database connectivity.
- CO4 Apply layout management and multi layout techniques to create adaptable user interface.
- CO5 Develop user interface for mobile application using widgets with event handling.
- CO6 Design pushes notifications for incoming messages.

IV. COURSE CONTENT:

1. Getting Started Exercises

1.1 Html layouts and links

- Create a webpage with HTML describing your department use paragraph and list tags.
- Apply various colors to suitable distinguish key words, also apply font styling like italics, underline and two other fonts to words you find appropriate, also use header tags.
- Create links on the words e.g. “Wi-Fi” and “LAN” to link them to Wikipedia pages.

1.2 Web Application design formatting

- Develop a web application with background banner image and navigation menus.
- Develop a web application using left menu(list).
- Develop setting to change the theme of entire web Application.

1.3 Login page

Design the following static Login page.

Username :	<input type="text"/>
Password:	<input type="password"/>
<input type="button" value="submit"/>	<input type="button" value="Reset"/>

1.4 Catalogue page

Design the following static web pages required for an online book store web site.

The catalogue page should contain the details of all the books available in the web site in a table.

- Branch
- Snap shot of Cover Page
- Author Name & Publisher.
- Price.
- Cart

1.5 Registration page

Design the following static web pages required for an online Registration from .

- E-mail id (Text field)
- First name (Text field)
- Last name (Text field)
- Phone number (text field)
- Gender (radio button)
- Date of birth (3 select boxes)
- Password (password filed)
- Retype the password (password filed)
- Submit query(button)
- Reset (button)

Exercises on java script

2.1 Validate page

Create a dynamic web page to validate the following fields.

- Name (Name should contains alphabets and the length should not be less than 6 characters).
- Password (Password should not be less than 6 characters length).
- E-mail id (should not contain any invalid and must follow the standard pattern name@domain.com)
- Phone number (Phone number should contain 10 digits only).

3. Online Recruitment System

Build a Online Recruitment System with the following functionalities:

1. **Administrator:** The information of the admin of the system is stored in this entity. It is stored data of login and the password. This provides security to the system and keeps the record of which user entered in the system at what instance of time.

It has the following attributes

- a. **Username:** It stores the name of the admin which acts as the unique name given to the manager of the firm.
 - b. **Password:** This attribute holds the secured keyword given to every manager of the educational institutions who need access to the system
 - c. **Login-Time:** The login time of the admin will be recorded in this field which helps in tracking the admin performance.
 - d. **Logout-Time:** It stores the logout time of the admin from the system
2. **Job Seeker:** To store the information of the candidate who registers to the system for participation in the recruitment process, the Job Seeker entity is created. It will have all the required data about the applicant.

It has following attributes

- a. **Can-Id:** Candidate Id is a unique number given to each candidate who registers in the system
 - b. **Can-Name:** Candidate Name is the information required for details of the applicant who registers in the system.
 - c. **Resume/CV:** In this attribute job, seekers can store their resumes which can be reviewed by the organization if they want to further check the candidate
 - d. **Projects:** Every candidate has many projects which they have created or worked on, which they want to share with them, recruiters.
 - e. **Personal-Info:** This attribute holds all the information the recruiters needed from the candidate apart from its technical skills and eligibility.
 - f. **Contact:** This stores the contact information of the candidate like phone number and email id
3. **Skill:** It is an essential part of the system as it works on both from recruiter and job seeker.

It has the following attributes

- a. **Skill-Id:** As every set of skills is unique so every set is given a unique number that acts as its id in the system.
 - b. **Technical:** It will store the technical skill by the applicant.
 - c. **Academic-details:** It will store the detailed information of the student in an academic field like high school marks, senior secondary marks. It will also have the graduation percentage is necessary for the skill.
 - d. **Co curricular-activities:** Every skill does not only require the technical talent of the student it also needs other social skills.
4. **Requisition:** In the recruitment management system, every employer will make the requisition of the application for the given post. From the application received from the number of the job seekers, the company will check their eligibility and decide whether he/she is compatible for the post or not. Then the shortlisted student will go for further rounds of the recruitment process as per the employer.

It has following attributes

- a. **Req-Id:** Every requisition present in the system given by each employer has unique numbers assigned to it.
- b. **Emp-Id:** This attribute will hold the information of the employer who is giving this requisition.
- c. **Post:** Every requisition given by the employer belongs to a specific post
- d. **Skill-Id:** It is a reference to a skilled entity that implies the required skill set for the specific post.
- e. **Package:** This attribute stores the information regarding the pay scale or salary
- f. **Criteria:** In the criteria attribute, data regarding the process of recruitment is stored.

5. **Employer:** It will store the detailed information of the employer present or registered in the system. Thus, making it one of the essential entities in the system. Before applying for any post, the applicant also wants to know about the company or the employer who has given the requisition.

It has the following attributes

- a. **Emp-Id:** It is the unique number given to the employer to differentiate from others.
 - b. **Emp-Name:** It will hold the name of the employer which is registered in the system.
 - c. **Information:** This entity will hold the information of the employer like when it is founded, what kind of projects they are working on
 - d. **Key-Persons:** It holds the data of the important persons of the company and the point of the contact for the interested candidates.
6. **Interview:** It has information about the interviews that will happen for the specific requisition. As the interview is an important part of any recruitment process as in this the employer and candidates come face to face and they can judge each other.

It has following attributes

- a. **Int-id:** It is the unique number given to each interview process that happens in the system
 - b. **Int-type-id:** As there are many kinds of interviews that can happen during the recruitment process.
 - c. **Req-Id:** It will hold the reference of the requisition for which a specific employer is conducting this interview.
 - d. **Remarks:** This has the remarks of the interviews which is given by the employers.
 - e. **Emp-Id:** It holds the data of the employer who is conducting this interview and managing the whole process.
7. **Interview-Type:** As there is various type of interview process which happens in the recruitment like technical for checking the technical knowledge of the candidate or the HR interview to test the social and interpersonal skill of the applicant.

It has following attributes

- a. **Int-type-id:** It is the unique number given to each type of interview
 - b. **Int-type-Name:** It holds the name of the type of interview.
8. **Offer-Letter:** When candidates apply for the job and it takes every criterion provided for a specific post. If it qualifies for the post the company will provide him/her with the offer letter. It is a document that makes the offer to the candidate for joining the employer's firm. It has detailed information regarding the package and the work environment of the company.

It has following attributes

- a. **Of-id:** Each offer letter is given a unique and distinct number
 - b. **Emp-Id:** It is the number given to the employer as it gives the information about which company has given this letter.
9. **Experience:** To show the talent each applicant writes the experience they have gain in the previous years. It helps the employer to see which applicant has the relevant knowledge required for the post. It is a weak entity dependent on the job seeker's profile. So, they can hire a person who has more potential than others.

It has following attributes

- a. **Exp-Details:** It holds the information candidate has learned during the work.
 - b. **Exp-Organization:** The name of the organization in which the applicant has gained the experience.
10. **Feedback:** A good system is where everyone has a say in everything. So, the feedback system is an entity that holds the feedback of the job seekers about their recruitment process for an employer or a requisition.

It has the following attributes:

- a. **Feed-Id:** It is a unique number given to each feedback. It makes the remark distinct and identifiable for the employers to take note of.
- b. **Emp-id:** It is a reference to the employer for which this feedback is given
- c. **Feed-details:** It holds the remark given by the applicant

See also for authentic understanding, click the link

<https://www.tracker-rms.com;>

[https://www.bullhorn.com/glossary/online-recruitment-system/;](https://www.bullhorn.com/glossary/online-recruitment-system/)

[https://www.manatal.com/;](https://www.manatal.com/)

<https://www.whatishumanresource.com/> and so on..

Each student has to refer any one of the web sites stated above.

4. Student Counseling Management System

Build a Student Counseling Management System with the following functionalities

1. **Student:** This entity will store the record of the student registered for the counseling. It is essential for the system as we need a separate unit which has all the information regarding the student and their personal and professional data. So, it will have the 10th and 12th percentage, rank in the entrance exam will also be asked from student to give the preferences according to it.

It has the following attributes:

- a. **Name:** This will store the name of the student as it needed while saving the required
- b. **DOB:** The date of birth of the student as mention in the high school certificate and will act as the candidate key in the database model
- c. **Student-id:** It is a unique alphanumeric number assigned to every student who registers for the counseling
- d. **Password:** It is an alphanumeric field that has the constraint of having a minimum of 8 digits and at least one alphabet with numbers
- e. **10th Percentage:** This is used to get the student's percentage in high school which some universities check before giving admission to the student.
- f. **12th Percentage:** It is an academic field that requires the student to give information about the 12th standard.
- g. **Rank-Entrance-Exam:** This attribute gets the entrance exam rank from the student which they have got according to their performance in the exam

2. **Choices:** The entity is needed to get the preferences from the student based on their rank in the entrance exam. The number of choices given to the student can be multiple based on the institution that requires the seats available for the courses. These preferences will go to the courses entity which then checks which selection is eligible for the student, this evaluation will be based on the number of seats available and the rank of the student.

This is also will be referenced in the allotment of the seats where the allotted seats are recorded.

It has the following attribute:

- a. **Rank-Entrance-Exam:** This attribute gets the entrance exam rank from the student which they have got according to their performance in the exam.
- b. **1st Preference:** This will be a simple attribute that holds the first preference of the student which he/she will select from the pool of choices.
- c. **2nd Preference:** It will hold the second preference of the student as it is not always possible to get the first selected choice as it may have got filled or the student with a higher rank got that seat.

3. **Courses:** It will have all the courses provided by the institutes. All the courses will have institute code will differentiate if the same courses are provided by the different institution. Every Course has the define the seats available and the allotted seats along with the total seats available which will provide transparency in the system.

It has the following attribute:

- a. **Course-code:** A unique number given to every course not depends on which university to it belongs, it helps in maintaining the record unique and distinct so act as the primary key for the entity.

- b. Institute-code:** Every institute that has register itself for the system of counseling has given a unique code that differentiates it from other institutes which have the same courses.
 - c. Total-seats :** It will have the total seats available for that course provided by the Institute.
 - d. No-of-seats:** It is a composite attribute which has the further information about a number of seats allotted and the vacant seat left for the course.
4. **Course-Name:** This entity will contain the name of the course and the unique id of that course. As the many institutes provide the same course it would be a difficult job to assign a different code to the same course. So, this field will be normalized and created a new entity out of it.
- It has the following attribute:**
- a. Course-code:** A unique number is given to every unique course.
 - b. Course-name:** The name of the course will be stored in this attribute.
5. **Allotment of Seat:** This entity will store the information of the student after the seat is confirmed for allotment.
- It has the following attribute:**
- a. Student-id:** Student unique id number.
 - b. Rank-Entrance-exam:** Rank gave to the student as per his/her performance in the entrance exam
 - c. Preference-no.:** The preference which got allotted, as per the choices are given to him.
 - d. Course-code:** The course code which got selected for.
 - e. Institute-code:** The institute allotted the seat.
6. **Institutes:** This entity will store the information regarding the institutes which are participating in the counseling. Each one of them has to publish their courses for which they want students with the number of the total seats.
- It has the following attributes:**
- a. Institute-code:** Every institute that has register itself for the system of counseling has given a unique code that differentiates it from other institutes which have the same courses.
 - b. Institute-name:** It will hold the name of the institute as mention by them in the registration process.
 - c. City:** It depicts the city to which the institute belongs to as it helps as a candidate key.
 - d. Password :** It is an alphanumeric field that has the constraint of having a minimum of 8 digits and at least one alphabet with numbers.
7. **Admitted-Student:** Once the student got allotted the seats and it gets confirmed by them Institute maintains their own record of which student has admitted to their institute and the courses what courses they have opted for.
- It has the following attributes:**
- a. Student-id:** Student unique id number.
 - b. Course-code :** The course code which got selected for.
 - c. Institute-code:** The Institute for which the seat is allotted. Institute is related to this entity as every university will have a database of how many students have been admitted to its courses.

See also for authentic understanding, click the link

<https://collegedunia.com/exams/cuet/counselling>; <https://www.academiaerp.com/>;
<https://nchmcounselling.nic.in/>, https://www.cdac.in/index.aspx?id=edu_acts_Login_cv11
 and so on..

Each student has to refer any one of the web sites stated above..

5.Data Mart Management System

Build a Data Mart Management system with the following functionalities

1. Master maintenance
2. Shipment
3. Billing
4. Reports

1. Master Maintenance :

Data Mart Management System maintains the following master details for various purposes.

It has the following attributes

- a. Suppliers Details
- b. Sub location In-charges Details
- c. Retailers Details
- d. Products Details

2. Shipment :

Data Mart Management System provides you a number of functions to maintain the flow of goods in warehouse.

It has the following attributes

- a. Transfer inventory
- b. Put inventory on hold
- c. View inventory balances
- d. View inventory transactions

3. Billing :

Warehouse billing process allows the retailer to generate new bills and also allows viewing of existing bills.

4. Reports :

The reports process allows the user to produce the reports necessary for day to day warehouse operations. The standard reports are ...

- a. Inventory Reports
- b. Purchase Order Reports
- c. Administrative Reports

See also for authentic understanding, click the link

<http://www.datamartsys.com/>; <https://netresultsgroup.com/>;
<https://www.indiamart.com/>; <https://uniondatamart.com/> and so on..
Each student has to refer any one of the web sites stated above.

6. Restaurant Reservation and Table Management Solutions

Restaurant Reservation and Table Management system is a Web application designed to help you (and your coworkers) to select the restaurant you are going to eat in. You can manage users, restaurants, menus, prices, give notations to each lunch, etc.

2. Reservation Management

- a. Easily enter or modify reservations while viewing guest histories.
- b. Capture phone numbers, email and mailing addresses.
- c. Allow management blocking and VIP pre-assignments.
- d. Reduce no-shows with enhanced customer tracking.
- e. Take reservations from your website or Open Table 24 hours.

3. Table Management

- a. Maximize seat utilization with walk-in and waitlist functionality.
- b. Instantly track covers for more efficient kitchen and server management. Increase table turns by tracking party status. Store multiple reservation sheets for holidays and special events.
- c. Hold and combine tables for large parties.
- d. Record and view shift notes for each day

2. Guest Management

- a. Identify regulars and VIPs
- b. Track customer preferences to meet and anticipate special requests
- c. View customer reservation histories at-a-glance
- d. Track special occasions such as guest birthdays and anniversaries Marketing Management
- e. Conduct powerful email marketing campaigns to increase repeat business.
- f. Print mailing labels to reach select target audiences.
- g. Track and reward concierge business.

3. Increase control

- a. Manage reservations from the back-office or any other location. Simultaneously control multiple restaurants from key centralized locations.
- b. Share guest data across sister restaurants.

See also for authentic understanding, click the link

<https://www.elluminatiinc.com/restaurant-reservation-system/>;
<https://restaurant.opentable.com/>; <https://www.tornosubitodubai.com/>;
<https://indiarestaurant.co.in/> and so on..

Each student has to refer any one of the web sites stated above

7. Secure Stock Exchange System using Web Services

This project implements a stock exchange is simply a system that is designed for the sale and purchase of securities of corporations and municipalities. A stock exchange sells and buys stocks, shares, and other such securities. In addition, the stock exchange sometimes buys and sells certificates representing commodities of trade.

The system “Secure Stock Exchange System using Web Services” consists of 3 modules

1. Stock Markets & Investments
2. Stock Options
3. Related Information

1. Stock Markets & Investments

- a. Stock Exchange Listing
- b. Stock Options & Analysis
- c. Stock Market Crash
- d. Selling Stock Certificates
- e. Stock Market Forecasts

2. Stock Options

Types of Stocks

- a. Stock Option Valuation
- b. Restricted Stock Options

3. Related Information

- a. Day Trading Stocks
- b. Stock Quotes & Stock Ticker
- c. Stock Charts
- d. Share Portfolio Management

See also for authentic understanding, click the link

<https://stocksandsecurities.adityabirlacapital.com/>; <https://www.nseindia.com/>;
<https://www.jpj.co.jp/english/>; <https://www.mstock.com/> and so on..

Each student has to refer any one of the web sites stated above.

8. Country Cargo and Express Couriers

This project is aimed at developing a data entry system that will enable the image assisted data entry for the cargo for the courier logistics. The scanned image if available will be displayed on the left side of screen when the Load ID is entered on the right side.

Module Description

1. The system should have appropriate login facility with relevant option like new user.
2. The system should allow admin to provide login id and passwords for the users with role
3. privilege option.
4. For appropriate user login the role privilege (Admin/Supervisor/User) should be selected.
5. The system should provide the Supervisor to set the priorities for the shipment load deliveries, also with services like ADD, UPDATE, VIEW, and DELETE on the shipment load item details.
6. Here the user is categorized as sender (source station) and receiver (destination station).
7. The system should provide the user (sender) to enter all his item details for shipment delivery.
8. The receiver after receiving the shipment delivery will be provided with the options of Viewing load details and also at the same time sends an act to both user (sender) and supervisor.

See also for authentic understanding, click the link

<https://worldwideexpresscouriers.com/> ; <https://www.vxpress.in/>;
<https://www.tciexpress.in/>; <https://www.bluedart.com/> and so on..
Each student has to refer any one of the web sites stated above

9. Food ordering application

Build a food ordering application with the following functionalities:

1. A 'Welcome page' which displays the logo and/or name of the app.
2. A 'Login Page' which asks for users' mobile number and password.
3. A 'Registration Page' which enables users to sign up for the app.
4. A 'Forgot Password Page' which enables users to reset their password.
5. A 'Navigation Drawer' with the app logo and user name on top and menu options to open the following pages:
 - a. Home
 - b. User Profile
 - c. Favourite Restaurants
 - d. Order History
 - e. Frequently Asked Questions (FAQs)
 - f. Log out
6. A 'My Profile' page (where the user's name, phone number, and address is displayed).
7. A 'Favorites' page (where the list of all favourite restaurants is displayed).
8. An 'Order History' page which lists the previously placed orders of the user.
9. An 'FAQ' page which lists some frequently asked questions. You can create any random questions which you feel could be relevant for a food delivery application. (Min. 5 questions)
10. A 'Logout' functionality which takes the user to the login page.
11. A 'Restaurant Details' page which displays the menu items of that particular restaurant, each item's price and the option to add an item to cart.
12. A 'Cart' page which lists the items added to cart and the total amount to be paid.

See also for authentic understanding, click the link

<https://www.swiggy.com/>; <https://www.zomato.com/deliver-food/>;
<https://www.ubereats.com/>, <https://www.dineout.co.in> and so on..
Each student has to refer any one of the web sites stated above.

10. Music player application

Build a music player application with the following functionalities:

1. A 'Splash screen' (gradient background and app logo in center)
2. A 'Navigation drawer' with app logo section at the top along with links to 'All Songs', 'Favourites', 'Settings' and 'About Us'.
3. An 'All songs' screen (where of list all the tracks fetched from offline storage are displayed and user can sort the tracks by name or recently added). This will be the home screen of the app.
4. The app should be able to fetch and play .mp3 and .wav files.
5. A 'Favourites' screen (where list of all the favourite songs are displayed)
6. A 'Settings' screen (where the 'Shake to change song' feature can be enabled or disabled)
7. An 'About us' screen (where we will display information about the app developer and the app version)
8. A 'Now playing' screen with following features:
 - a. Track title and track artist
 - b. Play / Pause button
 - c. Next button
 - d. Previous button
 - e. Shuffle button
 - f. Loop button
 - g. Seek bar
 - h. Mark track as favourite or unfavourite it

- i. Third party visualiser in upper half background
 - j. A 'Back to list' button in the header which should take the user to the screen he came from (kind of like back button behaviour).
 - k. Shake to change song
9. A 'Now playing' bar at the bottom with name of the track playing and play or pause feature. This would appear if the user has moved from 'Now playing' screen to 'All songs' screen or 'Favourites' screen without pausing the track.
 10. Background play. The app will continue playing the track if the app gets closed (not killed) without the music being paused.
 11. A notification saying "A track is playing in the background" only if the app gets closed (not killed) without the music being paused.
- Primary color scheme: #9b2a58, #00032a

See also for authentic understanding, click the link

<https://open.spotify.com>; <https://www.hungama.com>;

<https://wynk.in/music> and so on..

Each student has to refer any one of the web sites stated above.

11. Smart Health Prediction

Build Android Smart Health Prediction application which can be used by all patients or their family members who need help in emergency with the following functionalities:

This application comprises of 3 major modules with their sub-modules:

1. A 'User page' with menu options to open the following pages:
 - a. A 'Patient Login page' which displays patient Login to the application using his ID and Password.
 - b. A 'Patient Registration page' which asks if patient is a new user, he will enter his personal details and he will user Id and password through which he can login to the application.
 - c. A 'My Details page' which patient can view his personal details.
 - d. A 'Disease Prediction page' patient will specify the symptoms caused due to his illness. Application will ask certain question regarding his illness and application predict the disease based on the symptoms specified by the patient and application will also suggest doctors based on the disease.
 - e. A 'Search Doctor page' which the patient can search for doctor by specifying name, address or type.
 - f. A 'Feedback page' which the patient will give feedback this will be reported to the admin.
 - g. A 'Notification page' which the user needs to enter his/her prescription and then the user need to select a time on which he/she wants to receive notification. The application will send the notification regarding the prescribed medicine to the user on their device.
2. A 'Doctor page' with menu options to open the following pages:
 - a. A 'Doctor Login page' will access the application using his User ID and Password.
 - b. A 'Patient Details page' can view patient's personal details.
 - c. A 'Patient's Previous Details page' will get all information about patient's previous case history. That will help him to serve him better.
3. A 'Admin page' with menu options to open the following pages:
 - a. A 'Admin Login page' can login to the application using his ID and Password.
 - b. A 'Add Doctor page' can add new doctor details into the database.
 - c. A 'Add Disease page' can add disease details along with symptoms and type.
 - d. A 'View Doctor page' can view various Doctors along with their personal details.
 - e. A 'View Disease page' can view various diseases details stored in database.
 - f. A 'View Patient page' can view various patient details who had accessed the application.
 - g. A 'View Feedback page' can view feedback provided by various users.

See also for authentic understanding, click the link

<https://open.spotify.com>; <https://www.hungama.com>;

<https://wynk.in/music> and so on..

Each student has to refer any one of the web sites stated above

12. Hostel Management Application

Develop the hostel management application which will help to manage the hostel. The hostel managers can keep track of the hostellers' in and out timings and their daily entries. This system is intended to help hostel admin by allowing them to save student records and information about their rooms. It helps the admin from the manual work from which it is very difficult to find the record of the students.

This application comprises of following functionalities:

1. A 'Admin page' with menu options to open the following pages:
 - a. A 'Manage Rooms' page which the admin can choose to add general rooms or bedrooms to the system. If the admin chooses a bedroom, they can add beds. They can add, update or delete rooms.
 - b. A 'Manage Students' page which the admin can view, add, update or delete students from the system. They can allocate rooms to the students. Also, they can view the attendance of a student.
 - c. A 'View Attendance' page which the admin can filter by date AND room OR student-id. They can view all the student's attendance.
2. A 'Student page' with menu options to open the following pages:
 - a. A 'Login' page which the student can log in to the system using a username and password.
 - b. A 'Profile' page which the student can add or update their profile details.
 - c. A 'Change Password' which they can also change their password to the new one.
 - d. A 'Home' page the student can scan the QR Code and a log will be added. They can view their room allocation details.
 - e. A 'View Attendance' page the students are able to view only their attendance, they can also filter it by date.
3. A 'Guardian page' which the ward wish to go out from campus for any purpose, the guardian will login through the registered login then the guardian will send the permission request to warden's android phone by specifying the purpose and also duration. After successful sent, the message is available in the warden login for further action.
4. A 'Warden page' after receiving the guardian's message, warden has a choice with options approve and reject. Based on the purpose of out campus, Warden Selects approve or reject. If approves then the message with the ward details are sent to security check which indicates the request made by the guardian is permitted.

See also for authentic understanding, click the link

<https://www.hostelsnap.com;>
<http://www.ifnoss.com/edu/college-university-software/hostel-management-system.aspx;>
<https://www.ezeetechsys.com/absolute;> <https://www.resbird.com> and so on..
 Each student has to refer any one of the web sites stated above

13. Stay safe women security application

Build stay safe women security project is used to provide highly reliable security system for the safety of women. The proposed system is based upon advanced sensors and GPS. The basic aim of the system is to develop a low cost solution for GPS based women tracking system (Women Safety System). The main objective of the system is to track the current location of the person which has an android enabled mobile by extracting the longitude and latitude of that target person.

This application comprises of following functionalities:

1. A 'Scream Alarm' page used perfect for the females as well as other users that need some kind of safety alarm in case they found out that someone is following or stalking them. It also consists of two other types of scream alarm. It's an initial distraction which will buy some time and allow the user to escape from the trouble.
 - a. Male voice scream
 - b. Police siren.

The user could select one of his/her choice from the "Settings" of the application, as keeping in mind the two other scream alarms are also added in this application as nowadays safety and security is everybody's concern.

2. A 'Fake Call Timer' page which the fake call timer allows the user to make fake calls in the time of need. It helps user to escape from an undesirable situation citing an important call from anyone who needs him/her urgently and rest depends upon user creativity. This feature also helps the user to escape from boring social events

In order to make a fake call the user have to select the "Fake Call" icon and after that user could write any

name from which he/she wants a fake call. User could also set up the timer as per the requirement. The user could also set the default timer from the “Settings” icon of the application.

In a critical situation, the user just have to long term press the fake call button and automatically get a fake call as per the desired selected timer in the settings.

3. A ‘Where Are You’ which is used to find track friend. While first request is send by the sender. The sender will have to select the “Where Are You” icon and then a new dialog box of “Pick a Friend” will open up. The sender could select any friend and the request will be sent to the receiver. The receiver will accept that request from their end and a message will be sent to the receiver with the present location of the user.
4. A ‘Track Me’ which will track the user to view the exact dynamic location of the victim. First user have to send the Track Me request at the receivers end. The receiver will accept the request and then his/her name will appear on the friends you are tracking on the bottom of the application. The user could select that friend from there and then it will get automatically re-directed to the Google maps from where the user could view the exact location of the victim and also where’s he/she heading to.
5. A ‘Friends List’ page which shows all the contact numbers of family and friends which are added by the user through contacts. This could be done by selecting the contact icon on the bottom right corner of the friends list.
6. A ‘Settings’ page which consists of the following features -:
 - a. A ‘Emergency Services’ page allows the Stay Safe Application to send emergency notifications and SMS with the exact location to the emergency contacts.
 - b. A ‘Low Battery Alert’ page alert feature allows the Stay Safe Application to send low battery alert and SMS to the emergency contacts.
 - c. A ‘Set Scream Sound’ page which the user could select any scream sound as per the requirement.
 - d. A ‘Fake Call Timer (On Long press)’ page which the user could set the fake call default timer as per the requirement.
7. A ‘Emergency Distress Signal (SOS)’ which the distress signal will be generated by the user in case of an emergency. In order to generate the distress signal the user have to shake up his/her phone, then a distress signal will appear at the user end with a default timer of 5 sec. In the end distress signal will be sent to the emergency contacts added by the user at the time of registration. The application send SMS and user details as well as the exact location of the user through a push notification at the receiver end, before sending a distress signal the user first have to turn on the emergency services from the settings of the application.

When user launches the application in his/her Android phone, the very first screen which lands is the Login Screen. First the user have to register himself by entering the details as the respective name and contact number of the user.

See also for authentic understanding, click the link

<https://womensafetywing.telangana.gov.in;>

<https://nirbhayaapp.com;> <https://safetipin.com>;

<https://wsww.smart24x7.com> and so on..

Each student has to refer any one of the web sites stated above

14. Controlling Anti Ragging Application

Controlling anti ragging system facilitate college students to register a criticism in opposition to ragging immediately. Students would be required to log in, after which they would be capable to register their complaints. The grievance will then be dispatched to the worried authorities for well-timed action and the motion will be initiated immediately. Previous archives reveal that well timed action was once taken in every case, which in flip resulted in a fall in such cases. Now, with the launch of the app, the Ministry hopes to wipe out the exercise completely.

This application comprises of following functionalities:

1. A ‘User Module’ page which the person can register with the machine then he/she will get the get entry to in the app. After login done if the any ragging is happened then without delay update the details. Every other flexibility is additionally on hand right here to contact the emergency wide variety also provided.
2. A ‘College Admin’ page which can operate the action on the failed complaints; complaints can be considered via the admin and can take the perfect action on the complaint.
 - a. A ‘Add Complaint’ page users can a make complaint against ragging. In this, those who are filling complaint as to fill some fields like Accused Name, Ragging Type, Mobile Number

- b. A 'View Complaint' page which can view the complaints made by students.

See also for authentic understanding, click the link

<http://www.amanmovement.org>;
<https://www.nmc.org.in/ActivitiWebClient/open/initiateAntiRaggingHome>;
<https://www.antiragging.in>; <https://www.amanmovement.org/raggingmain.html> and so on..
Each student has to refer any one of the web sites stated above.

15. Extracurricular Event Tracking Application

Build Extracurricular activities are sources that have long been a part of the educational system; students participate in these activities, which are not part of the standard curriculum or teaching techniques. Participation in all of these activities, or even just one of them has been linked to social and academic success. Students who engage in extracurricular activities gain from the numerous options available to them. Having better grades, higher standardized test scores and educational attainment, attending school more consistently, and having a greater self-concept were all advantages of participating in extracurricular activities.

Our Extracurricular Event Tracking System is developed to track students' hours organizing and participating in cultural events and college fests. This system keeps students updated about upcoming events and maintains track of the events they have attended and the hours they have spent organizing various cultural programs. Thereby, encouraging participation.

This application comprises of following functionalities:

1. A 'Admin' page which contains the following features:
 - a. A 'Login' page which the admin can log in using their credentials.
 - b. A Manage Course which they can add, update, view and delete courses. The courses will be BSC/MSc/Diploma/Engineering, etc. They can add total hours.
 - c. A 'Manage Students' page which the admin can add, update, view and delete students. They can assign courses to the students.
 - d. A 'Manage Events' page which the admin can add, update, view and delete events. They can add details about a particular event. They can choose courses. They can add the link to the Meet.
 - e. A 'View Records' page which to view the records, the admin can choose by course. They can view the students' list and list of records.
2. A 'Student' page which contains the following features:
 - a. A 'Login' page which the students can log in using their credentials.
 - b. A 'Dashboard' page which the students can see the hours completed or the total hours. They can see up to 5 upcoming events ordered by date and time.
 - c. A 'Calendar' page the students can see the Day/Week/Month on the Calendar.
 - Day: The system will show today's events on the date. The user can also select any previous date to check any events that happened on that day.
 - Week: By selecting any week, the system will show one week's dates & events.
 - Month: By selecting the month, the system will show the current month's dates and events.
 - d. A 'Records' page which the system will show all the events attended.
 - e. A 'Profile' page which the user can view their profile and change the password.
 - f. A 'Forgot Password' page which if the user forgets their password, an OTP will be sent via Email or Phone number (Any 1). They would require to enter the correct OTP and reset the password.
 - g. A 'Notification' page which the system will send a notification to the user before an event starts, and the dashboard checks the event unattended or hours not completed.

See also for authentic understanding, click the link

<http://www.amanmovement.org>;
<https://www.nmc.org.in/ActivitiWebClient/open/initiateAntiRaggingHome>;
<https://www.antiragging.in>; <https://www.amanmovement.org/raggingmain.html> and so on.

Each student has to refer any one of the web sites stated above

16. Student management system

Build a student management system will have data on every student, and check the daily attendance of every student. The system comprises 3 major modules with their sub-modules as follows:

1. The 'Admin' page menu options to open the following pages
 - a. A 'Login' page the admin can log in to the system using a username and password.
 - b. A 'Manage Students' page the admin can manage student details and they can add, update and delete details. Also, add a fingerprint while adding the student's details.

- c. A 'Manage Teacher' page admin will also add teacher's details and add, update and delete details. Teachers will be assigned subjects.
- d. A 'Manage Grades' page can manage students' grades and add, update or delete grades.
- e. A 'Manage Subjects' page can also manage subjects by adding, updating and deleting subjects.
- f. A 'Manage Appointments' page applies to meet parent and waits for a reply.
- g. A 'Appointments' page parents applied to meet the teacher/admin. The admin will respond to the parent's appointment request.
- h. A 'Manage Leaves' page able to manage pending and past student leave. They can view a list of all the applied student's leaves. The admin can view all the pending leaves applied by parents. The admin will approve or reject the leave application. Leave Note: They will be able to send a note to the parent related to leaves.
- i. A 'Medical Certificate' page will be able to view the medical certificate submitted by parents. They can send a note with an acknowledgement in response.
- j. A 'View Complaint' page can view pending and replied complaints. They will be able to respond to the complaints.
- k. A 'Warning Letters' page contains the admin can add and view a list of all students with attendance below 80% / total course work below 60%. They can send a warning letter to their parents. Admin will be able to view all warning letters and replies by parent.
- l. A 'Attendance' page contains Entry and Exit Lecture-wise Attendance. See the attendance list of all Students grade-wise against 2 dates.
- m. A 'Reports' page contains can view reports of all the student's attendance lists, grade-wise. They can view students' attendance reports for 2,4,6-months with lecture, entry and exit details. They can also view the academic reports of all the students.
- n. A 'Entry/Exit' page can add entry and exit of the student biometric.

2. A 'PARENTS' page contains following features

- a. A 'Login' page can log in to the system using login credentials.
- b. A 'Profile' page can add or update details to their profile.
- c. A 'Change Password' can also change their old password to the new one.
- d. A 'Forget Password' used if they forget their login password, they can receive a link to change the password on their registered email-id.
- e. A 'Appointments' page contains the parent can apply to meet the admin/teacher and waits for their response. The parent will also receive the admin's request to meet them and they can respond to the request.
- f. A 'Leaves' page contains Parents can view all the leaves including pending leaves. They can view the list of all their applied leaves. Parents can view leaves that have not been replied to by an admin.
They can also cancel their applied leaves.
- g. A 'Medical Certificate' page contains Add: Parents can upload the medical reports of their child while applying for leave. They can view all the previously uploaded lists and replies from the admin.
- h. A 'Complaint' page contains Parents can add a new complaint. They can view a list of all the complaints and replies from the admin.
- i. A 'Warnings' page list of the warning letters sent by admin and parents can respond to them.
- j. A 'Attendance' page can view their child's attendance, lecture-wise. Also, they can view the attendance list of the child against 2 dates.
- k. A 'Reports' page contains following features: Academic attendance reports can be viewed by the admin. Parents can view attendance reports of 2/4/6 Months with lecture, entry and exit details. They can also view their child's academic report.
- l. A 'Notifications' page will receive notifications such as when appointment status changes or admin applied, leaves approved/rejected, medical certificate reply, complaint reply, warning letter, student entry/exit, academic marks uploaded by teacher.

3. A 'TEACHER' page contains following features

- a. A 'Login' page can log in to the system using a username and password.
- b. A 'Profile' page can add or update details to their profile.
- c. A 'Change Password' can also change their passwords.
- d. A 'Forget Password' if they forget their login password, they can receive a link to change the password on their registered email-id.
- e. A 'My Subjects' page contains following features
Assigned Subjects: A list of all assigned subjects can be viewed grade-wise.
Students List: The teacher can view the list of the students in that Grade.

- m. A 'Attendance' page contains they can select grade and subject to view attendance. Biometric Attendance: They can take the biometric attendance of students. See the Attendance list of all Students grade-wise against 2 dates, lecture-wise.
- n. A 'Academics' page contains teachers can choose the grade, subject and student for their academic reports. They can view, add and update students' academic details. Teachers can add marks on tests, assignments and final exams. They can update existing marks on tests, assignments and final exams.
- o. A 'Complaint' page contains teachers can add new student complaint. They can view list of all the complaints and replies from admin.

See also for authentic understanding, click the link

<https://www.iitms.co.in/products/student-information-system-sis>,

<https://samvidha.iare.ac.in>;

<https://themeforest.net/item/clever-course-learning-management-system-theme/8645312>;

<https://markerspro.in> and so on.

Each student has to refer any one of the web sites stated above

17. Pharm easy application

Build a pharm easy application for ordering medicines and tests. This application comprises of following functionalities

1. A "Home Page" which consists of a list medicines & health care product.
2. A "Lab Tests page" contains different medical tests.
3. A "Health Care Products page" contains vitamins and necessary to our body.
4. A "Offers Page" offers on different products and coupons.
5. A "Account Page" contains details of individual customer.
6. A "Products Page" contains a list of products.
7. A "Detailed View of Product" detailed view of selected product.
8. A "Cart page" list of items we selected to buy.
9. A "Delivery Details Page" consists of customer address and contact details.
10. A "Payment Page" how to make payments (cod or online payments).
11. A "Success page" show transaction success.
12. A "Bottom APP Bar" consists of logos which opens the corresponding pages(fragments).
 - a. Home page
 - b. Test Page
 - c. Health Care Page
 - d. Offers Page
 - e. Account Page

See also for authentic understanding, click the link

<https://pharmeasy.in>; <https://www.netmeds.com>;

<https://www.medlife.com>; <https://www.practo.com> and so on.

Each student has to refer any one of the web sites stated above.

18. News Application

Build a news application is to connect news articles from all around the world and deliver it to user as fast as possible in best visualize way. This application comprises of following functionalities:

1. A 'User Interface' page should be able to select from different categories, countries and newspaper. Short News as list view with header, little description and image before showing full article can be helpful to user to determine what type of news they are looking for. View Holder can be used for this list view for better and fast experience.
2. A 'Admin Panel' page controls the User and Writers logins from database. Writers can add news, update and delete from its database as per required. Main Admin can add Users, Writers, and News. He can also approve, update and delete it. Using this approach, we can create network in local areas connect by writers and local admins which will provide news at local level and we can also implement location feature which will update local news of different location or city.
3. A 'Global Support' page contains different type of newspaper will be available from all around the world in different languages with this user will be able to get news from all around the world.
4. A 'Short News' page News will be displayed in short format with title, image and little description in list view. It will help user to access required news faster.

5. A 'Search Option' page which user will be able to search from not only one source but many different sources available within API.
6. A 'Favorites / Offline Reading' page which News can be added as favorites which will automatically will be saved for offline reading.
7. A 'Sharing' page which user will be able to share news easily on social media.

See also for authentic understanding, click the link

<https://indianexpress.com>; <https://www.bbc.com/news>;

<https://www.ndtv.com>; <https://www.hindustantimes.com> and so on..

Each student has to refer any one of the web sites stated above.

V. TEXT BOOKS:

1. Thomas A. Powell, "*The Complete Reference*", "HTML and CSS", 5th Edition, 2017
2. Elisabeth Robson , Eric Freeman, "*Head First HTML and CSS: A Learner's Guide to Creating Standards-Based Web Pages*", 2nd Edition, 2012
3. Adam Boduchand Roy Derks, "*React and React Native: A Complete Hands-on Guide to Modern Web and Mobile Development with React.js*", 3rd Edition, 2020.
4. RetoMeier, "*ProfessionalAndroid 4 Application Development*", 1st Edition, Wile Publication

VI. REFERENCE BOOKS:

1. W Hans Bergsten, "*Java Server Pages*", O'Reilly, 3rd Edition, 2003.
2. D. Flanagan, "*Java Script*", O'Reilly, 6th Edition, 2011.
3. Jon Duckett, "*Beginning Web Programming*", WROX, 2nd Edition, 2008.
4. Bill Phillips and Chris Stewart, "*Android Programming*", The Big Nerd Ranch Guide, 3rd Edition, 2017.
5. Dawn Griffiths, David Griffiths, "*Head First Android Development: A Brain-Friendly Guide*", 2017.
6. Antonio Leiva , Kotlin for Android Developers: Learn Kotlin while developing an Android App, Create Space Independent Publishing, 2016

VII. ELECTRONICS RESOURCES:

1. <https://www.codecademy.com/learn/paths/web-development/>
2. <https://nptel.ac.in/courses/106/105/106105084/>
3. <https://medium.com/@aureliomerenda/create-a-native-web-app-with-react-native-web-419acac86b82>
4. <https://www.coursera.org/learn/react-native>
5. <https://desirecourse.net/react-native-and-redux-course-using-hooks>
6. <https://www.javatpoint.com/android-tutorial>
7. <https://www.tutorialspoint.com/android/index.htm>
8. <https://docs.flutter.dev/>
9. <https://developer.android.com/courses/android-basics-compose/course>
10. <https://developer.android.com/guide>

VIII. MATERIALS ONLINE

1. Course template
2. Lab Manual



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

GENDER SENSITIZATION								
II Semester: AE / ME / CE / ECE / EEE / CSE / CSE (CS) / CSE(DS) / CSE (AI & ML) / IT								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AHSD10	Mandatory	L	T	P	C	CIA	SEE	Total
		-	-	-	-	-	-	-
Contact Classes: Nil	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: Nil			
Prerequisite:								

I. COURSE OVERVIEW:

The course aims at raising awareness of gender equality among students from sociological, cultural, psychological, legal and economical perspectives, thereby empowering them to communicate better in a cross-cultural work ambience. At the end of this course the students expose to better egalitarian interactions between men and women and to enable them see diversity and inclusiveness as assets in a globalized scenario.

II. COURSES OBJECTIVES:

The students will try to learn

- The basic gender concepts and their application to the long- term implementation of programming and initiatives.
- The gender roles, expectations and issues and their impact on as the community's day-to-day life.
- The more egalitarian interactions between men and women.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO 1 Develop a better understanding of important issues related to gender in contemporary India.
- CO 2 Sensitize to the basic dimensions of the biological, sociological, psychological and legal aspects of gender.
- CO 3 Acquire insight into the gendered division of labor and its relation to politics and economics.
- CO 4 Attain a finer grasp of how gender discrimination works in our society and how to counter it.
- CO 5 Men, women and professionals will be better equipped to work and live together as equals.
- CO 6 Know the new laws that provide protection and relief to women.

IV. COURSE CONTENT:

MODULE-I: UNDERSTANDING GENDER

Introduction: definition of gender, basic gender concepts and terminology, exploring attitudes towards gender, construction of gender. Socialization: making women, making men, preparing for womanhood. growing up male, first lessons in caste.

MODULE – II: GENDER ROLES AND RELATIONS

Two or many; struggles with discrimination; Gender roles and relations: types of gender roles, gender roles and relationships matrix, missing women, sex selection and its consequences, declining sex ratio, demographic consequences; Gender Spectrum: Beyond the Binary.

MODULE-III: GENDER AND LABOUR

Division and valuation of labour; Housework: the invisible labor, my mother doesn't work. Share the load work: its politics and economics, fact and fiction. unrecognized and unaccounted work; gender development issues gender, governance and sustainable development; gender and human rights; gender and mainstreaming.

MODULE-IV: GENDER AND BASED VIOLENCE

The concept of violence; types of gender; based violence, gender, based violence from a human Rights perspective, Sexual harassment: say no sexual harassment, not eve-teasing-, coping with everyday harassment, further reading, chupulu. Domestic violence: speaking out is home a safe place, when women unite (Film). rebuilding lives, thinking about sexual violence blaming the victim, i fought for my Life.

MODULE–V: GENDER AND CULTURE

Gender and film; gender and electronic media; gender and advertisement; gender and popular, literature, Gender development issues, gender issues, gender sensitive language, gender and popular literature. Just relationships: being together as equals; Mary kom and onler, love and acid just do not mix, love letters, mothers and fathers, Rosa Parks, the brave heart.

V. TEXT BOOKS:

1. A. Suneetha, Uma Bhargubanda, Duggirala Vasanta, Rama Melkote, Vasudha Nagaraj, Asma Rasheed, Gogu Shyamala, Deepa Sreenivas and Susie Tharu The Textbook, *Towards a World of Equal: A Bilingual Textbook on X Gender*, published by Telugu Academy, Telangana Government, 1st Edition, 2015.

VI. REFERENCE BOOKS:

1. Kadambari V, *Gender Studies: A Primer*. Rajiv Gandhi National Institute of Youth Development, Sriperumbudur. 1st Edition, 2009.
2. C. Rajya Lakshmi Kalyani, D.S. Vittal, A. Kanaka Lakshmi, P. Chandrakala, B. Lavanya., *Gender Sensitization*, Himalaya Publishing House. 1st Edition, 2017.

VII. ELECTRONICS RESOURCES:

1. <https://en.unesco.org/women-make-the-news-2017/resources>
2. http://ncw.nic.in/sites/default/files/Booklet-%20Gender%20Sensitization_0.pdf



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

PROBABILITY AND STATISTICS								
III Semester: AE / ME / CE / CSE / CSE(AI&ML) / CSE(DS) / CSE(CS) / IT								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AHSD11	Foundation	L	T	P	C	CIA	SEE	Total
		3	1	-	4	40	60	100
Contact Classes: 48	Tutorial Classes: 16	Practical Classes: Nil			Total Classes: 64			
Prerequisite:								

I. COURSE OVERVIEW:

Probability theory is the branch of mathematics that deals with modelling uncertainty. The course includes: Bay's theorem, random variables, probability distributions, hypothesis testing, confidence interval and linear regression. The use of probability models and statistical methods is for analyzing data, designing, manufacturing a product and the observed class frequencies for engineering and sciences.

II. COURSES OBJECTIVES:

The students will try to learn

- The theory of random variables, basic random variate distributions and their applications.
- The Methods and techniques for quantifying the degree of closeness among two or more variables and the concept of linear regression analysis.
- The Estimation statistics and Hypothesis testing which play a vital role in the assessment of the quality of the materials, products and ensuring the standards of the engineering process.
- The statistical tools which are essential for translating an engineering problem into probability model.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO 1 Explain the probability elementary theorems on probability conditional, multiplication, Baye's theorem under randomized probabilistic conditions.
- CO 2 Apply the role of random variables and types of random variables, expected values of the discrete and continuous random variables under randomized probabilistic conditions
- CO 3 Apply the parameters of random variable Probability distributions such as Binomial, Poisson by using their probability functions,
- CO 4 Interpret the parameters of random variate Probability distributions such as Binomial, Poisson and Normal distribution by using their probability functions, expectation and variance
- CO 5 Make Use of estimation statistics in computing confidence intervals by Correlation Analysis, Regression analysis
- CO 6 Identify the role of statistical hypotheses, types of errors, confidence intervals, the tests of hypotheses for large and small sample, in making decisions over statistical claims in hypothesis testing.

IV. COURSE CONTENT:

MODULE-I: PROBABILITY (10)

Probability, axiomatic approach, elementary theorems on probability, conditional probability, multiplication theorem, Bayes theorem (without proof).

MODULE-II: RANDOM VARIABLES (09)

Random variables: Discrete and continuous random variables, probability distribution, probability mass function and probability density function.

MODULE-III: PROBABILITY DISTRIBUTION (10)

Binomial distribution: Mean and variance of Binomial distribution, Poisson distribution: Poisson distribution as a limiting case of Binomial distribution, mean and variance of Poisson distribution.

Normal distribution: mean, variance, mode, median of Normal distribution.

MODULE-IV: CORRELATION AND REGRESSION (09)

Correlation- Karl Pearson's coefficient of correlation, rank correlation, repeated ranks. Regression: Lines of regression, regression coefficient, angle between two regression lines.

MODULE-V: TEST OF HYPOTHESIS (10)

Population, Sample, standard error; Test of significance: Null hypothesis, alternate hypothesis. Types of errors, level of significance.

Large sample tests: Test of hypothesis for single mean, difference between means, single proportion and difference between proportions. Small sample tests: Student's t-distribution, F-distribution and Chi-square distribution.

V. TEXT BOOKS:

1. Erwin Kreyszig, "Advanced Engineering Mathematics", John Wiley & Sons Publishers, 9th Edition, 2014.
2. B. S. Grewal, "Higher Engineering Mathematics", Khanna Publishers, 42nd Edition, 2012.

VI. REFERENCE BOOKS:

1. S. C. Gupta, V. K. Kapoor, "Fundamentals of Mathematical Statistics", S. Chand & Co., 10th Edition, 2000.
2. N. P. Bali, "Engineering Mathematics", Laxmi Publications, 9th Edition, 2016.
3. Richard Arnold Johnson, Irwin Miller and John E. Freund, "Probability and Statistics for Engineers", Prentice Hall, 8th Edition, 2013.

VII. ELECTRONIC RESOURCES:

1. <http://e4uhu.com/down/Applied/9th>
2. <https://toaz.info/32fa2f50-8490-42cf-9e6a-f50cb7ea9a5>
3. <http://www.mathworld.wolfram.com>

VIII. MATERIALS ONLINE

1. Course template
2. Tutorial question bank
3. Tech talk topics
4. Open end experiments
5. Definitions and terminology
6. Assignments
7. Model question paper - I
8. Model question paper - II
9. Lecture notes
10. E-learning readiness videos (ELRV)
11. Power point presentation

COURSE CONTENT

MECHANICS OF SOLIDS								
III Semester: AE								
Course Code	Category	Hours /Week			Credits	Maximum Marks		
AAED01	Core	L	T	P	C	CIA	SEE	Total
		3	0	-	3	40	60	100
Contact Classes: 48	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: 48			
Prerequisite: Engineering Mechanics								

I. COURSE OVERVIEW:

Mechanics of solids focuses on analyzing stresses and deflection in deformable solids and structural components subjected to different types of forces and thermal loadings. It allows for the safe design of structures that are capable of supporting their intended loads, and also decide either it is suitable for a particular application by satisfying the safety and serviceability conditions. This course introduces the concepts of simple stresses, strains, and principal stresses and focuses on the analysis of members subjected to axial, shear, bending, and torsional loads. In a nutshell, the course aims at developing the skill to solve engineering problems on strength of materials. It acts as a pre-requisite to the advanced courses on Aircraft structures and Analysis of aircraft structures.

II. COURSES OBJECTIVES:

The students will try to learn:

- The concepts of mechanics of deformable solids and their constitutive relations (including stress – strain relations), principal stresses and strains, and resilience produced under various loading conditions used in predicting the strength of aircraft structures.
- The procedure of estimating shear force - bending moment, twisting moment, flexural Stresses, shear stresses in beams subjected to various loadings, for designing the shape, size and material of aircraft components.
- The methods for determining the slope and deflection of beams and critical load on columns subjected to various loading conditions for determining the stiffness and strength of aircraft structures.
- The methods of failures and distribution of stresses in cylinders due to internal pressure.

III. COURSE OUTCOMES:

After successful completion of the course, students should be able to:

- CO 1 Demonstrate the concepts of stress-strain, material constitutional relationship and strain energy induced in isotropic materials under different loadings.
- CO 2 Make use of the concepts of shear force and bending moment in beams, and power transmission in shafts for analyzing the strength under different loadings.
- CO 3 Apply theory of distribution of bending stresses and shearing stress across the beams for the safe design of aircraft components subjected different loadings.
- CO 4 Utilize Maxwell's reciprocal theorem and double integration for determining the slope and deflections in beams under different types of loadings.
- CO 5 Utilize Euler's formula for determining the buckling load in columns under different end conditions.
- CO 6 Apply the concepts of longitudinal and circumferential stresses induced in cylinders for the safe design under inside and outside pressure.

IV. COURSE CONTENT:

MODULE – I: STRESSES AND STRAINS (09)

Elasticity and plasticity – Types of stresses and strains–Hooke’s law– stress – strain diagram for mild steel – Working stress – Factor of safety – Lateral strain, Poisson’s ratio, volumetric strain – Elastic module and the relationship between them – Bars of varying section – composite bars – Temperature stresses. Strain energy – Resilience – Gradual, sudden, impact and shock loadings.

MODULE – II: SHEAR FORCE AND BENDING MOMENT (09)

Definition of beam – Types of beams – Concept of shear force and bending moment – S.F and B.M diagrams for cantilever, simply supported and overhanging beams subjected to point loads, u.d.l., uniformly varying loads and combination of these loads – Point of contra flexure – Relation between S.F., B.M and rate of loading at a section of a beam.

TORSION OF CIRCULAR SHAFTS: Theory of pure torsion – Derivation of Torsion equations: $T/J = q/r = C\theta/L$ – Assumptions made in the theory of pure torsion – Torsional moment of resistance – Polar section modulus – Power transmitted by shafts;

MODULE – III: FLEXURAL STRESSES AND SHEAR STRESSES (09)

Theory of simple bending – Assumptions – Derivation of bending equation: $M/I = f/y = E/R$ Neutral axis – Determination bending stresses – section modulus of rectangular and circular sections (Solid and Hollow), I,T,Angle and Channel sections – Design of simple beam sections.

Derivation of formula – Shear stress distribution across various beams sections like rectangular, circular, triangular, I, T angle sections

MODULE –IV: DEFLECTION OF BEAMS AND COLUMNS (09)

Deflection in simply supported beams and cantilever beams with concentrated loads, uniformly distributed loads and their combination using Double integration method and Macaulay ‘s method,

Columns and Struts:Introduction; Failure of a column; Euler’s column theory; assumptions in the Euler’s column theory; Sign conventions; types of end conditions of the columns; columns with both ends hinged; column one end fixed and other end free; columns with both ends fixed; columns with one end fixed and other end hinged; Euler’s formula and equivalent length of a column; slenderness ratio; limitations of Euler’s formula, Empirical formulae for columns; Rankine’s formulae; Johnson’s formula for columns; Indian Standard code for columns

MODULE –V: PRINCIPAL STRESSES, STRAINS AND CYLINDERS (09)

Introduction – Stresses on an inclined section of a bar under axial loading – compound stresses – Normal and tangential stresses on an inclined plane for biaxial stresses – Two perpendicular normal stresses accompanied by a state of simple shear – Mohr’s circle of stresses – Principal stresses and strains – Analytical and graphical solutions. Thin Cylinders: Thin seamless cylindrical shells – Derivation of formula for longitudinal and circumferential stresses – hoop, longitudinal and volumetric strains – changes in diameter, and volume of thin cylinders due to internal pressure; Thick Cylinders: Lamé’s equation- cylinders subjected to inside and outside pressures

V. TEXT BOOKS:

1. R. K Bansal, “Strength of Materials”, Laxmi publications, 6th edition, 2018.
2. T. H. G. Megson, “Aircraft Structures for Engineering Students”, Butterworth-Heinemann Ltd, 6th edition, 2015.
3. Gere, Timoshenko, “Mechanics of Materials”, McGraw Hill, Reprint 2020.

VI.REFERENCE BOOKS:

1. B. C. Punmia, Ashok K Jain and Arun K Jain, “Mechanics of Materials”, Laxmi Publications Pvt. Ltd., New Delhi, 12th edition, 2007.

VII.ELECTRONICS RESOURCES:

1. http://www.efunda.com/sm_home/sm.cfm
2. <https://nptel.ac.in/courses/112105171/1>

VIII.MATERIALS ONLINE:

1. Course template
2. Tutorial question bank
3. Tech talk topics
4. Open end experiments
5. Definitions and terminology
6. Assignments
7. Model question paper - I
8. Model question paper - II
9. Lecture notes
10. E-learning readiness videos (ELRV)
11. Power point presentation



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

THERMODYNAMICS AND HEAT TRANSFER								
III Semester: AE								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AAED02	Core	L	T	P	C	CIA	SEE	Total
		3	0	-	3	40	60	100
Contact Classes: 48	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: 48			
Prerequisite: Nil								

I. COURSE OVERVIEW:

Engineering thermodynamics deals with the study of energy and heat transfer in various engineering applications. This course will delve into several key areas, including the principles of mass and energy conservation, the application of the first law to both closed and open systems, a comprehensive grasp of the second law of thermodynamics and its relation to entropy, an exploration of pure substance properties, a thorough examination of power generation and refrigeration. It forms an essential cornerstone for mechanical, chemical and aerospace engineers and plays a pivotal role in the study of energy systems.

II. COURSE OBJECTIVES:

The students will try to learn:

- The concepts of thermodynamics, gas properties and the thermodynamic disorderness in the real time physical systems such as heat engines, heat pumps
- The characteristics of pure substances, mixtures, usage of steam tables, Mollier' chart and psychometric charts for solving thermal problems.
- The principles of various power cycles, gas compressors and their real-world applications.
- The various modes of heat transfer, types of heat exchangers

III. COURSE OUTCOMES:

After successful completion of the course, students will be able to:

- CO 1 Interpret the thermodynamic processes and energy conversions in physical systems based on fundamental laws of thermodynamics for identifying the significance of energy.
- CO 2 Make use of heat to work conversion and thermodynamic direction laws involved in heat engines and heat pumps for deriving their efficiency and coefficient of performance
- CO 3 Utilize thermodynamic laws and entropy to describe the properties of pure substances and mixtures of perfect gases for examining the unavailability in any given system.
- CO 4 Choose the properties of refrigerants and practicing of psychometric charts for solving the complex problems in refrigeration and air conditioning.
- CO 5 Illustrate the working principles of air standard cycles and its performance characteristics for recognizing the suitable engines in aeronautical and automobile applications
- CO 6 Summarize the basics of heat transfer, working principle of heat exchangers for relating their applications in aerospace engineering.

IV. COURSE CONTENT:

MODULE-I: BASIC CONCEPTS AND FIRST LAW OF THERMODYNAMICS (10)

Basic concepts: System, control volume, surrounding, boundaries, universe, types of systems, macroscopic and microscopic viewpoints, concept of continuum, thermodynamic equilibrium, state, property, process, cycle, reversibility, quasi static process, irreversible process, causes of irreversibility, various flow and non-flow, energy in state and in transition, types-work and heat, point and path function, Zeroth law of thermodynamics, concept of quality of temperature, Principles of thermometry,

reference points, constant volume gas thermometer, ideal gas scale, PMMI Joule's experiments, first law of thermodynamics, corollaries first law applied to a process, applied to a flow system, steady flow energy equation.

MODULE –II: SECOND LAW OF THERMODYNAMICS (09)

Limitations of the first law: thermal reservoir, heat engine, heat pump, parameters of performance, second Law of thermodynamics, Kelvin Planck and Clausius statements and their equivalence, Corollaries, PMM of second kind, Carnot's principle, Carnot cycle and its specialties, thermodynamic scale of temperature, Clausius inequality, Entropy, principle of Entropy increase, availability and irreversibility, thermodynamic potentials, Gibbs and Helmholtz functions, Maxwell relations, Third Law of thermodynamics.

MODULE –III: PURE SUBSTANCES AND MIXTURES OF PERFECT GASES (10)

Pure substances: Phase transformations, T-S and H-S diagrams, P-V-T surfaces, triple point at critical state properties during change of phase, dryness fraction, Mollier charts, psychometric properties, dry bulb temperature, wet bulb temperature,

Dew point temperature, thermodynamic wet bulb temperature, specific humidity, relative humidity, saturated air, vapour pressure, degree of saturation, adiabatic saturation, Carrier's equation, Psychometric chart.

MODULE –IV: POWER CYCLES AND GAS COMPRESSORS (09)

Power cycles: Otto, Diesel, Dual combustion cycles, description and representation on P-V and T-S diagram, thermal efficiency, mean effective pressures on air standard basis, comparison of cycles, introduction to Brayton cycle, Basic concepts of: Gas Compressors, Types of Air Compressors, Single-Stage compression, Multistage Compression, Volumetric Efficiency, Rotary Compressors.

MODULE –V: BASIC CONCEPTS IN HEAT TRANSFER (10)

Modes of Heat Transfer, Heat Transmission by Conduction, Fourier's law of conduction Thermal conductivity of materials Thermal resistance, General heat conduction equation in cartesian coordinates, Heat conduction through plane and composite walls, Heat Transfer by Convection, Heat Exchangers, Types of heat exchangers Heat exchanger analysis Logarithmic temperature difference (LMTD), Heat Transfer by Radiation, Surface emission properties, Absorptivity, reflectivity and transmittivity, Concept of a black body, The Stefan-Boltzmann law, Kirchhoff's law.

V. TEXT BOOKS:

1. P. K. Nag, "Engineering Thermodynamics", Tata McGraw-Hill, 4th edition, 2008.
2. YunusCengel, Michael A. Boles, "Thermodynamics-An Engineering Approach", Tata McGraw-Hill, 7th edition, 2011.
3. R.K.Rajput, "Engineering Thermodynamics", Laxmi Publications (P) Ltd, 3 rd edition, 2007.

VI. REFERENCE BOOKS:

1. J. B. Jones, R. E. Dugan, "Engineering Thermodynamics", Prentice Hall of India Learning, 1st edition, 2009.
2. Y. V. C. Rao, "An Introduction to Thermodynamics", Universities Press, 3rd edition, 2013.
3. K. Ramakrishna, "Engineering Thermodynamics", Anuradha Publishers, 2nd edition, 2011.
4. Holman. J.P, "Thermodynamics", Tata McGraw-Hill, 4th edition, 2013

VII. ELECTRONICS RESOURCES:

1. <https://www3.nd.edu/~powers/ame.20231/planckdover.pdf>
2. <http://www.ebookdownloadz.net/2014/08/engineering-thermodynamics-by-pknag.html>

VIII. MATERIALS ONLINE

1. Course template
2. Tutorial question bank
3. Tech talk topics

4. Open end experiments
5. Definitions and terminology
6. Assignments
7. Model question paper – I
8. Model question paper - II
9. Lecture notes
10. E-learning readiness videos (ELRV)
11. Power point presentation

COURSE CONTENT

FLUID DYNAMICS								
III Semester: AE								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AAED03	Core	L	T	P	C	CIA	SEE	Total
		3	0	-	3	40	60	100
Contact Classes: 48	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: 48			
Prerequisite: Nil								

I. COURSE OVERVIEW:

Fluid dynamics is the study of fluids either in motion or at rest. This course introduces to a broad range of fundamental concepts, methods of fluid mechanics, mathematical description of fluid flows and the solution of some important flow problems. The course emphasizes importance of dimensionless numbers in various engineering fluid flow problems. It discusses the concept of boundary layer theory and bluff body aerodynamics. Compare and contrast various fluid machinery based on flow properties and its applications.

II. COURSE OBJECTIVES:

The students will try to learn:

- The fundamental knowledge of fluids, their properties and behavior under various conditions of closed conduit and external flow systems.
- Various mathematical models in fluid dynamics, how they are derived and how they can be used to solve practical problems
- The importance of formation of boundary layer when fluid flows over the solid bodies and effect in reduction of displacement, momentum, energy and pressure gradient.
- Working principle of various turbo machineries, their application and analyze their characteristics using governing equations.

III. COURSE OUTCOMES:

After successful completion of the course, students will be able to:

- CO 1 Identify the suitable pressure measuring devices for determining the flow measurements in fluid systems
- CO 2 Utilize the concept of Similitude and Non-Dimensional numbers for validating physical parameters of a designed prototype
- CO 3 Apply the law of conservation of mass and momentum for obtaining numerical solutions of internal fluid flow systems.
- CO 4 Utilize the principle of Bernoulli equation for measurement of discharge in internal and external fluid flow systems
- CO 5 Apply boundary layer theory for internal and external flow systems in determining drag forces and frictional losses.
- CO 6 Classify the types of hydraulic machines based on working principle and performance characteristics for the selection in real world applications.

IV. COURSE CONTENT:

MODULE-I: FLUID PROPERTIES AND FLUID STATICS (10)

Density, specific weight, specific gravity, surface tension and capillarity, Newton's law of viscosity, incompressible and compressible fluid, numerical problems; Hydrostatic forces on submerged bodies - Pressure at a point, Pascal's law, pressure variation with temperature and height, center of pressure plane, vertical and inclined surfaces, Manometers - simple and differential Manometers, inverted

manometers, micro manometers, pressure gauges and numerical problems. Buoyancy - Archimedes principle, metacenter, Meta centric height calculations; Stability.

MODULE –II: DIMENSIONAL ANALYSIS (10)

Fundamental and secondary quantities, Dimensional homogeneity, Methods of dimensional Analysis-Rayleigh's method, Buckingham's π - theorem, method of selecting repeating variables, similarity parameters - Reynolds number,

Froude number, Euler's number, Weber's number, Mach number concepts of geometric, kinematic and dynamic similarity

MODULE –III: KINEMATICS AND DYNAMICS OF FLUIDS (09)

Methods of describing fluid motion, types of fluid flows, differential form of continuity equation-Cartesian, cylindrical and polar coordinate system, Numerical problems.

Euler's equation of Motion; Bernoulli's equation, Application of Bernoulli's equation in flow measurements: velocity and mass flow rate, pitot-static tube, venturi meter, orifice meter and V-Notch.

MODULE –IV: BOUNDARY LAYER THEORY (10)

Introduction and classification of boundary layer, boundary layer properties Displacement, momentum and energy thickness, Boundary Layers with Pressure Gradient, laminar-boundary-layer equations, limitations, Shear Stress in a Boundary Layer, Blasius Solution for Flat-Plate Flow, drag force on flat due to boundary layer, idea of boundary layer separation, separation control, streamlined and bluff bodies.

MODULE –V: TURBO MACHINERY (09)

Introduction and classification of fluid machines: Turbo machinery analysis; The angular momentum principle; Euler turbo machine equation; Application to fluid systems, working principle of Pelton wheel, Francis turbine, Kaplan turbine, reciprocating pump, centrifugal pump, gas turbine, velocity triangle.

V. TEXT BOOKS:

1. Frank M. White, "Fluid Mechanics", McGraw Hill Education Private Limited, 9th edition, 2022.
2. R. K Bansal, "Fluid Mechanics and Hydraulic Machines", Laxmi publications ltd, 10th edition, 2019.
3. Yunus Cengel, John Cimbala, "Fluid Mechanics: Fundamentals and Applications", McGraw Hill Education Private Limited, 4th edition 2017.

VI. REFERENCE BOOKS:

1. Yuan S W, "Foundations of fluid Mechanics", Prentice-Hall, 2nd edition, 1987.
2. Milne Thompson L M, "Theoretical Hydrodynamics", MacMillan, 5th edition, 1968.
3. Som S. K, Biswas. G, "Introduction to fluid mechanics and fluid machines", Tata McGraw-Hill, 2nd Edition, 2004.

VII. ELECTRONICS RESOURCES:

1. <https://nptel.ac.in/courses/112105171/1>
2. <https://textofvideo.nptel.iitm.ac.in/112105171/lec1.pdf>

VIII. MATERIALS ONLINE

1. Course template
2. Tutorial question bank
3. Tech talk topics
4. Open end experiments
5. Definitions and terminology
6. Assignments
7. Model question paper – I
8. Model question paper - II

9. Lecture notes
10. E-learning readiness videos (ELRV)
11. Power point presentation

COURSE CONTENT

DATA STRUCTURES								
III Semester: AE / ME / CE / CSE / CSE (AI & ML) / CSE (DS) / CSE (CS) / IT / ECE / EEE								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
ACSD08	Foundation	L	T	P	C	CIA	SEE	Total
		3	0	-	3	40	60	100
Contact Classes: 48	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: 48			
Prerequisite:								

I. COURSE OVERVIEW:

The course covers some of the general-purpose data structures and algorithms, and software development. Topics covered include managing complexity, analysis, static data structures, dynamic data structures and hashing mechanisms. The main objective of the course is to teach the students how to select and design data structures and algorithms that are appropriate for problems that they might encounter in real life. This course reaches to student by power point presentations, lecture notes, and lab which involve the problem solving in mathematical and engineering areas.

II. COURSES OBJECTIVES:

The students will try to learn

- The skills needed to understand and analyze performance trade-offs of different algorithms / implementations and asymptotic analysis of their running time and memory usage.
- The basic abstract data types (ADT) and associated algorithms: stacks, queues, lists, tree, graphs, hashing and sorting, selection and searching.
- The fundamentals of how to store, retrieve, and process data efficiently.
- The implementing these data structures and algorithms in Python.
- The essential for future programming and software engineering courses.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO1 Interpret the complexity of algorithm using the asymptotic notations.
- CO2 Select appropriate searching and sorting technique for a given problem.
- CO3 Construct programs on performing operations on linear and nonlinear data structures for organization of a data
- CO4 Make use of linear data structures and nonlinear data structures solving real time applications.
- CO5 Describe hashing techniques and collision resolution methods for accessing data with respect to performance.
- CO6 Compare various types of data structures; in terms of implementation, operations and performance.

IV. SYLLABUS:

MODULE – I: INTRODUCTION TO DATA STRUCTURES, SEARCHING AND SORTING (10)

Basic concepts: Introduction to data structures, classification of data structures, operations on data structures; Algorithm Specification, Recursive algorithms, Data Abstraction, Performance analysis-time complexity and space complexity, Asymptotic Notation-Big O, Omega, and Theta notations. Introduction to Linear and Non Linear data structures, Searching techniques: Linear and Binary search; Sorting techniques: Bubble, Selection, Insertion, Quick and Merge Sort and comparison of sorting algorithms.

MODULE – II: LINEAR DATA STRUCTURES (09)

Stacks: Stack ADT, definition and operations, Implementations of stacks using array, applications of stacks, Arithmetic expression conversion and evaluation; Queues: Primitive operations; Implementation of queues using Arrays, applications of linear queue, circular queue and double ended queue (deque).

MODULE – III: LINKED LISTS (10)

Linked lists: Introduction, singly linked list, representation of a linked list in memory, operations on a single linked list; Applications of linked lists: Polynomial representation and sparse matrix manipulation.

Types of linked lists: Circular linked lists, doubly linked lists; Linked list representation and operations of Stack, linked list representation and operations of queue.

MODULE - IV NON LINEAR DATA STRUCTURES (09)

Trees: Basic concept, binary tree, binary tree representation, array and linked representations, binary tree traversal, binary tree variants, threaded binary trees, application of trees, Graphs: Basic concept, graph terminology, Graph Representations - Adjacency matrix, Adjacency lists, graph implementation, Graph traversals – BFS, DFS, Application of graphs, Minimum spanning trees – Prims and Kruskal algorithms.

MODULE - V BINARY TREES AND HASHING (10)

Binary search trees: Binary search trees, properties and operations; Balanced search trees: AVL trees; Introduction to M-Way search trees, B trees; Hashing and collision: Introduction, hash tables, hash functions, collisions, applications of hashing.

V. TEXT BOOKS:

1. Rance D. Necaise, “Data Structures and Algorithms using Python”, Wiley Student Edition.
2. Benjamin Baka, David Julian, “Python Data Structures and Algorithms”, Packt Publishers, 2017.

VI. REFERENCE BOOKS:

1. S. Lipschutz, “Data Structures”, Tata McGraw Hill Education, 1st Edition, 2008.
2. D. Samanta, “Classic Data Structures”, PHI Learning, 2nd Edition, 2004.

VII. ELECTRONICS RESOURCES:

1. https://www.tutorialspoint.com/data_structures_algorithms/algorithms_basics.htm
2. <https://www.codechef.com/certification/data-structures-and-algorithms/prepare>
3. <https://www.cs.auckland.ac.nz/software/AlgAnim/dsToC.html>
4. <https://online-learning.harvard.edu/course/data-structures-and-algorithms>

VIII. MATERIALS ONLINE

1. Course template
2. Tutorial question bank
3. Tech talk topics
4. Open end experiments
5. Definitions and terminology
6. Assignments
7. Model question paper – I
8. Model question paper - II
9. Lecture notes
10. E-learning readiness videos (ELRV)
11. Power point presentation



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

NUMERICAL METHODS USING MATLAB								
III Semester: AE								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AAED04	Core	L	T	P	C	CIA	SEE	Total
		-	-	2	1	40	60	100
Contact Classes: Nil	Tutorial Classes: Nil	Practical Classes: 45			Total Classes: 45			
Prerequisite: Nil								

I. COURSE OVERVIEW:

Numerical Methods using MATLAB involves the application of mathematical techniques to solve complex problems through numerical approximation. MATLAB, a powerful programming language and environment, is widely used for implementing and analyzing numerical methods. This approach is essential in fields such as engineering, physics, and finance, where analytical solutions may be challenging or impossible to obtain. The use of MATLAB facilitates efficient algorithm implementation, visualization of results, and quick prototyping of numerical solutions. Students and professionals benefit from learning and utilizing numerical methods in MATLAB to address real-world problems in a computational context.

II. COURSE OBJECTIVES:

The students will try to learn:

- I. The procedures, algorithms, and concepts require to solve specific problems.
- II. The concepts of algebra, calculus and numerical solutions using MATLAB software.
- III. The knowledge in MATLAB and can apply for project works.
- IV. The simple mathematical functions and operations thereon using plots/display.

III. COURSE OUTCOMES:

After successful completion of the course, students will be able to:

- CO1 Understand the numerical methods in MATLAB use for the accurate of solutions to mathematical problems that may lack analytical solutions or have complex expressions.
- CO2 Make use of MATLAB a user-friendly platform for implementing numerical algorithms efficiently, enabling the quick and reliable solution of mathematical problems.
- CO3 Utilize MATLAB's built-in plotting and visualization tools facilitate the interpretation and presentation of numerical results, aiding in a better understanding of the solution behavior.
- CO4 Apply Numerical methods often involve iterative processes, and MATLAB's programming capabilities make it easy to refine and optimize algorithms for improved convergence and accuracy.
- CO5 Make use of MATLAB's matrix-oriented approach is well-suited for handling large datasets, making it advantageous for numerical methods dealing with extensive data or complex systems of equations.
- CO6 Apply Numerical methods in MATLAB find applications in various disciplines, including engineering, physics, finance, and more, providing a versatile toolset for solving diverse computational challenges.

IV. COURSE CONTENT:

Exercises for Numerical Methods Using MATLAB

Note: Students are encouraged to bring their own laptops for laboratory practice sessions.

1. Getting Started with MATLAB Student Version

1.1 MATLAB Student Version Installation procedure

System requirement

Supported Platforms and Operating Systems:

Microsoft Windows 10, 64-bit

Minimum Hardware Requirements for MATLAB Student Product:

Processor(s): Workstation class

4 GB RAM

25 GB hard drive space

Computer must have a physical C:/ drive present

Graphics card and driver: Professional workstation class 3-D

OpenGL-capable

Installation Procedure

1. Extract (unzip) the downloaded installation files.
2. Right-click on setup.exe and select Run as Administrator. (This will run setup.exe from the extracted files.)
3. Read and accept the clickwrap to continue.
4. Click the right arrow button to accept the default values throughout the installation.
5. Click the exit button to close the installer.
6. The MATLAB Student software is now installed.
7. Reboot your machine and then run the MATLAB. Student product from your Start menu by selecting Workbench.

Problem size limits

- No Geometry Export

Limits for MATLAB Student and Discovery (Refine Mode)

- Aerospace Tool Box: 250K nodes/elements
- No. of Licensed units: 60

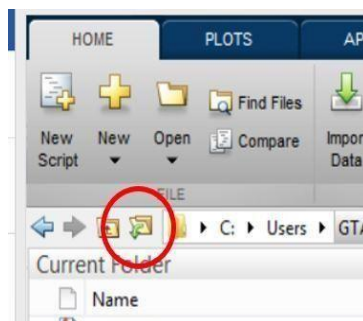
1.2 Getting Started with MATLABworkbench

Open MATLAB: Windows Start Menu button → MATLAB → Workbench • Under the Toolbox Analysis Systems category, click and drag analysis system Static Structural onto the Project Schematic, drop it on target Create standalone system (this will be the only target available)

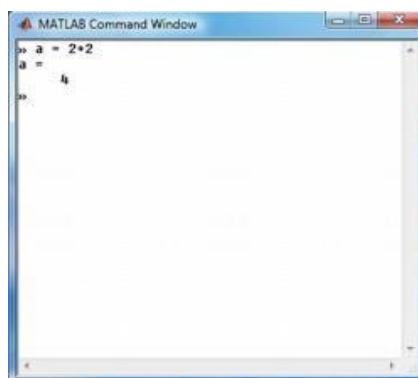


The desktop includes these panels:

Current Folder - This panel allows you to access the project folders and files.



Command Window - This is the main area where commands can be entered at the command line. It is indicated by the command prompt (>>).

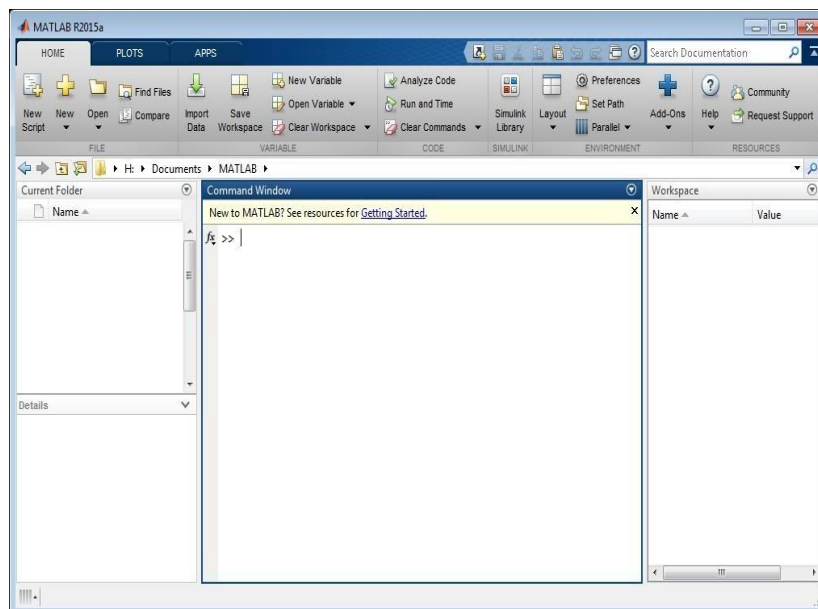


Command History - This panel shows or rerun commands that are entered at the command line.

```

end
edge
edgedetect
imshow(edge);
edgedetect
02-07-2013 02:27
13-07-2013 18:27
imread('home.jpg');
I=imread('home.jpg');
I=imread('home.jpg');
I=imread('home.jpg');
imshow(I);

```



You are now faced with the MATLAB desktop on your computer, which contains the prompt (>>) in the Command Window. Usually, there are 2 types of prompt:>>For full version
EDU> for educational version

Note:

1. To simplify the notation, we will use this prompt, >>, as a standard prompt sign, though our MATLAB version is for educational purpose.
2. MATLAB adds variable to the workspace and displays the result in the Command Window.

2. Exercises on Algebra

Algebra is a branch of mathematics that deals with symbols and the rules for manipulating those symbols. It typically involves solving equations and working with variables to analyze and understand mathematical relationships. Here are some key aspects of algebra. At an

advanced level, algebra becomes more abstract, studying structures like groups, rings, fields, and vector spaces. Abstract algebra generalizes and formalizes the fundamental algebraic concepts and techniques encountered in earlier study.

2.1 Roots of the equation

Calculate the a) Find the roots of the equations $6x^5 - 41x^4 + 97x^3 - 97x^2 + 41x - 6$

Hints

1. Roots of Equation $6x^5 - 41x^4 + 97x^3 - 97x^2 + 41x - 6$

```
• v = [6, -41, 97, -97, 41, -6]; % writing the coefficients
s = roots(v);
```

2. General Post Processing

```
• disp('The first root is: '), disp(s(1));
• disp('The second root is: '), disp(s(2));
• disp('The third root is: '), disp(s(3));
• disp('The fourth root is: '), disp(s(4));
• disp('The fifth root is: '), disp(s(5));
```

2.2 Solving the linear equations

A linear equation is an equation that involves only constants and variables raised to the first power, with no exponents or other operations.

Find the values of x, y, z of the equations $x+y+z=3, x+2y+3z=4, x+4y+9z=6$

Hints

1. Identify the Variables: Determine which letters represent the variables in the equation. Often, variables are represented by letters like x, y, and z.
2. Define the Coefficients Matrix and the Constants Vector: You need to represent your linear equations in matrix form $Ax=b$, where A is the coefficients matrix, xx is the variables vector, and bb is the constants vector.
3. Choose a Suitable Solver: MATLAB provides several solvers for different types of linear systems. The choice of solver depends on the properties of your matrix AA and the specific problem you're solving.

```
% Define the coefficients matrix A and the constants vector b
A = [2, 1; -1, 3];
b = [5; 10];
```

```
% Solve the linear system Ax = b using linsolve
x = linsolve(A, b);
```



```
% Display the solution
disp('The solution to the linear system is:');
disp(x);
```

Try

1. Solve a system of linear equations using MATLAB, that can represent the system in matrix form $Ax=b$, where A is the matrix of coefficients, x is the vector of unknowns, and b is the vector of constants
2. Solve a system of linear equations using MATLAB, with different forms by ODE forms.

3. Exercises on control structures

Control structures in MATLAB allow you to control the flow of execution of your code. They enable you to make decisions, repeat operations, and perform tasks conditionally. They are fundamental to writing effective and efficient MATLAB code, allowing you to implement complex logic and handle various scenarios dynamically.

3.1 Conditional statements allow you to execute specific blocks of code based on certain conditions (if, elseif, else Statements).

```
x = 10;
if x < 5
disp('x is less than 5');
elseif x == 5
disp('x is equal to 5');
else
disp('x is greater than 5');
end
```

Hints

```
% Example 1: Temperature evaluation using 'if' statement
temperature = 25;
if temperature > 30
disp('It is hot');
elseif .....
```

```
disp(.....);
else
disp(.....);
end
```

```
% Example 2: if-else statement
y = 3;

if y > 5
disp('y is greater than 5');
else
disp('y is not greater than 5');
```

```

end

% Example 3: if-else statement
y = 3;
if y > 5
    disp('.....');
else
    disp('.....');
end

```

3.2 Switch Case

The **switch** statement in MATLAB is used for executing one of several groups of statements, depending on the value of a variable or expression

The basic syntax of the switch statement in MATLAB is as follows:

```

switch expression
case value1
    % Code to execute if expression is equal to value1
case value2
    % Code to execute if expression is equal to value2
    % Add more cases as needed
otherwise
    % Code to execute if expression does not match any case
end

```

Hints

```

% Example 1: Simple switch Statement
day = 'Monday';

switch day
case 'Monday'
    disp('Start of the work week. ');
case .....
    disp('.....');
case .....
    disp('Weekend has started!');
case 'Sunday'

```

```

    disp('.....');
otherwise
    disp('Middle of the work week. ');
end

```

```

% Example 2: Switch with Numeric Values

number = 3;

switch number
case 1
    disp('One');
case .....
    disp('Two');
case .....

```

```

disp('.....');
case .....
disp('.....');
otherwise
disp('Number is not between 1 and 4');
end

% Example 3: Nested Switch Statements

shape = 'rectangle';
color = 'blue';

switch shape
case.....
disp('Shape is a circle. ');
case 'rectangle'
disp('Shape is a rectangle. ');
switch.....
case.....
disp('Color is red. ');
case 'blue'
disp('Color is blue. ');
otherwise
disp.....
end
otherwise
disp('Shape is unknown. ');
end

```

Try

1. Change the location of point load and calculate the shear force, bending moment and stress distribution in the beam
2. Change the location of point loads in tapered beam and find the deflection and the displacement variation along the beam length

3.3 FOR loop MATLAB

In MATLAB, a for loop is used to execute a sequence of statements multiple times, iterating over a range of values. The loop variable increments or decrements according to a specified range, and the block of code within the loop is executed once for each value in that range.

The basic syntax of a for loop in MATLAB is:

```

for index = startValue:endValue
    % Code to execute
end

```

Hints

```

% Example 1: A simple for loop that iterates from 1 to 5 and displays the
iteration variable

for i = 1:5
disp.....
end

```

```

% Example 2: Loop Through a Vector

vec = [10, 20, 30, 40, 50];

for i = .....
disp(['Element ', num2str(i), ' is ', num2str(vec(i))]);
end

% Example 3: Sum of First N Natural Numbers

N = 10;
sum = 0;

for i .....
sum.....
end

disp.....

```

4. Exercises on Matrices

4.1 Addition, Subtraction, and Multiplication of Matrices

Matrices are a fundamental aspect of MATLAB, which stands for "Matrix Laboratory". MATLAB is designed for matrix operations, making it an ideal environment for working with linear algebra, numerical analysis, and related fields.

Matrix addition is done element-wise, and the matrices must be of the same size.

```
A = [1 2 3; 4 5 6; 7 8 9];
```

```
B = [9 8 7; 6 5 4; 3 2 1];
```

```
C = A + B;
```

The result is:

```
C =
```

```
10 10 10
```

```
10 10 10
```

```
10 10 10
```

Hints

```

% Example 1: Matrix subtraction is done element-wise, and the matrices must
be of the same size.

A = [1 2 3; 4 5 6; 7 8 9];
B = [9 8 7; 6 5 4; 3 2 1];

D = A - B;

The result is:

```

```

D =
    -8    -6    -4
    -2     0     2
     4     6     8

% Example 2: For matrix multiplication, the number of columns in the first
matrix must equal the number of rows in the second matrix.

A = [1 2 3; 4 5 6];
B = [7 8; 9 10; 11 12];
E = A * B;

The result is:

E =
    58    64
   139   154

% Example 3: Element-wise Multiplication.

A = [1 2 3; 4 5 6];
B = [7 8 9; 10 11 12];

D = .....;

disp('.....');
disp(elementWiseProd);

The result is:

D .....

```

Try

1. Change the number of elements, matrix size and find the element wise multiplication and division.
2. Change the number of dimensions, matrix size and apply it to solve ODE problems.

5. Exercises on system of linear equations

5.1 Rank of the matrix

In MATLAB, the rank of a matrix is determined by the number of linearly independent rows or columns. The rank function is used to compute this value, which is essential in understanding the properties of the matrix, such as its solvability in linear systems and its inevitability.

Example 1: Rank of a Full Rank Matrix

```
A = [1 2 3; 4 5 6; 7 8 10];
```

```
r = rank(A);
```

```
disp(['The rank of the matrix A is ', num2str(r)]);
```

The output will be:

The rank of the matrix A is 3

Hints

```
% Example 1: Rank of a Matrix with Linearly Dependent Rows
B = [1 2 3; 2 4 6; 3 6 9];
r = rank(B);
```

```
disp(['The rank of the matrix B is ', num2str(r)]);
```

Output:

The rank of the matrix B is 1

```
% Example 2: Rank of a Rectangular Matrix
```

```
C = [1 2 3; 4 5 6; 7 8 9; 10 11 12];
r = rank(C);
disp(['The rank of the matrix C is ', num2str(r)]);
```

Output:.....

The rank of the matrix C is 2

```
% Example 3: Full Rank Matrix
```

```
A = [1 2 3; 4 5 6; 7 8 9];
r = rank(A);
disp.....
```

Output:.....

```
% Example 4: Linearly Dependent Rows
```

```
B = [1 2 3; 2 4 6; 3 6 9];
r = rank(B);
disp.....
```

Output:.....

```
% Example 5: Rectangular Matrix
```

```
C = [1 2 3; 4 5 6; 7 8 9; 10 11 12];
r = rank(C);
disp.....
```

Output:.....

```
% Example 6: Identity Matrix
```

```
I = eye(4);
r = rank(I);
disp.....
```

Output:.....

```
% Example 7: Sparse Matrix
```

```
D = [1 0 0; 0 0 0; 0 0 2];
r = rank(D);
disp.....
```

Output:.....

```
% Example 8: Random Matrix
```

```

E = rand(5, 5);
r = rank(E);
disp('.....')

Output:.....

% Example 9: Singular Matrix
F = [1 2 3; 4 5 6; 7 8 9];
r = rank(F);
disp('.....')

Output:.....

% Example 10: Zero Matrix
G = zeros(3, 3);
r = rank(G);
disp('.....')

Output:.....

```

Try

1. Use the matrix operations for rectangular matrices and find the rank of resultant matrix.
2. Matrix operations element wise for square matrices and find the rank of resultant matrix.

5.2 Row Echelon Form

To transform a matrix into its Row Echelon Form (REF) in MATLAB, you can use the built-in function `rref` which stands for "reduced row echelon form". However, if you want to implement the algorithm manually, here's a step-by-step guide and corresponding MATLAB code to achieve this.

Steps to Transform a Matrix to Row Echelon Form

1. Start with the leftmost nonzero column. This is a pivot column. The pivot position is at the top of this column.
2. Select the pivot row by finding the row with the largest absolute value in the pivot column. Swap this row with the top row if necessary.
3. Normalize the pivot row by dividing the entire row by the pivot element.
4. Eliminate all entries below the pivot position by subtracting an appropriate multiple of the pivot row from each row below.
5. Cover the row containing the pivot position and repeat the process on the submatrix that remains. Continue until there are no more pivot positions to cover

Syntax:

```
function REF = rowEchelonForm(A)
```

```

[rows, cols] = size(A);

REF = A; % Initialize REF with the matrix A

% Iterate over each column
for col = 1:cols

    % Find the pivot row
    pivotRow = col;

    while pivotRow <= rows && REF(pivotRow, col) == 0
        pivotRow = pivotRow + 1;
    end

    % If no pivot row is found, move to the next column
    if pivotRow > rows
        continue;
    end

    % Swap the current row with the pivot row
    if pivotRow ~= col
        temp = REF(col, :);
        REF(col, :) = REF(pivotRow, :);
        REF(pivotRow, :) = temp;
    end

    % Normalize the pivot row
    REF(col, :) = REF(col, :) / REF(col, col);

    % Eliminate all entries below the pivot
    for row = col + 1:rows
        REF(row, :) = REF(row, :) - REF(row, col) * REF(col, :);
    end
end
end
end

```

Sample Problem:

```
A = [1 2 3; 4 5 6; 7 8 9];
```

```
RREF = rref(A);
```

```
disp('Reduced Row Echelon Form:');
```



```
disp(RREF);
```

Hints

```
% Example 1: For matrix A = [1 2 3; 4 5 6; 7 8 9], modify it in the rowEchelonForm
```

```
Row Echelon Form:
```

```
1.0000    2.0000    3.0000
      0   -3.0000   -6.0000
      0         0         0
```

```
Using rref(A) will produce:
```

```
Reduced Row Echelon Form:
```

```
1.0000         0   -1.0000
      0    1.0000    2.0000
      0         0         0
```

```
% Example 2: Simple Matrix
```

```
A = [1 2 3; 4 5 6; 7 8 9];
```

```
REF_A = rowEchelonForm(A);
```

```
disp('Row Echelon Form (Custom Function) for A:');
```

```
disp(REF_A);
```

```
RREF_A = rref(A);
```

```
disp('Reduced Row Echelon Form (Built-in Function) for A:');
```

```
disp(RREF_A);
```

```
% Example 3: Rectangular Matrix
```

```
B = [1 2 3; 4 5 6; 7 8 9; 10 11 12];
```

```
REF_B = rowEchelonForm(B);
```

```
disp('Row Echelon Form (Custom Function) for B:');
```

```
disp(REF_B);
```

```
RREF_B = rref(B);
```

```
disp('Reduced Row Echelon Form (Built-in Function) for B:');
```

```
disp(RREF_B);
```

```
% Example 4: Matrix with Linearly Dependent Rows
```

```
C = [1 2 3; 2 4 6; 3 6 9];
```

```
REF_C = rowEchelonForm(C);
```

```
disp('Row Echelon Form (Custom Function) for C:');
```

```
disp(REF_C);
```

```
RREF_C = rref(C);
```

```
disp('Reduced Row Echelon Form (Built-in Function) for C:');
```

```
disp(RREF_C);
```

```
% Example 5: Sparse Matrix
```

```
D = [1 0 0; 0 0 0; 0 0 2];
```

```
REF_D = rowEchelonForm(D);
```

```
disp('Row Echelon Form (Custom Function) for D:');
```

```
disp(REF_D);
```

```
RREF_D = rref(D);
```

```
disp('Reduced Row Echelon Form (Built-in Function) for D:');
```

```
disp(RREF_D);
```

5.3 LU Decomposition

LU decomposition, also known as LU factorization, is a method of decomposing a matrix into the product of a lower triangular matrix LL and an upper triangular matrix UU. This is particularly useful for solving linear systems of equations, inverting matrices, and calculating determinants.

Syntax

```
function [L, U] = luDecomposition(A)

    % Get the size of the matrix
    [n, m] = size(A);

    % Initialize L and U
    L = eye(n);
    U = zeros(n);

    % Perform the LU Decomposition
    for j = 1:n
        % Upper Triangular Matrix U
        for i = 1:j
            U(i,j) = A(i,j) - L(i,1:i-1) * U(1:i-1,j);
        end

        % Lower Triangular Matrix L
        for i = j+1:n
            L(i,j) = (A(i,j) - L(i,1:j-1) * U(1:j-1,j)) / U(j,j);
        end
    end
end
```

Sample Problems

```
A = [4 3; 6 3];

% LU Decomposition using built-in function
[L, U] = lu(A);

disp('Matrix A:');

disp(A);

disp('Lower Triangular Matrix L:');
```

```

disp(L);

disp('Upper Triangular Matrix U:');

disp(U);

% Example Matrix

A = [4 3; 6 3];

% LU Decomposition using custom function

[L, U] = luDecomposition(A);

disp('Matrix A:');

disp(A);

disp('Lower Triangular Matrix L:');

disp(L);

disp('Upper Triangular Matrix U:');

disp(U);

```

Hints

```

% Define the matrix
A = [4 3; 6 3];

% Perform LU Decomposition using built-in function
[L, U] = lu(A);

% Display the results
disp('Matrix A:');
disp(A);
disp('Lower Triangular Matrix L:');
disp(L);
disp('Upper Triangular Matrix U:');
disp(U);

```

Output:

Matrix A:

```

4    3
6    3

```

Lower Triangular Matrix L:

```

1    0
1.5  1

```

Upper Triangular Matrix U:

```

4    3
0   -1.5

```

```
% Define the matrix
B = [1 2 3; 4 5 6; 7 8 9];

% Perform LU Decomposition using built-in function
[L, U] = lu(B);

% Display the results
disp('Matrix B:');
disp(B);
disp('Lower Triangular Matrix L:');
disp(L);
disp('Upper Triangular Matrix U:');
disp(U);
```

Output:

Matrix B:

```
1    2    3
4    5    6
7    8    9
```

Lower Triangular Matrix L:

```
1    0    0
0.1429    1    0
0.5714    0.5000    1
```

Upper Triangular Matrix U:

```
7    8    9
0    0.8571    1.7143
0    0    0
```

```
% Define the matrix
C = [4 3 2 1; 3 2 1 4; 2 1 4 3; 1 4 3 2];
```

```
% Perform LU Decomposition using built-in function
[L, U] = lu(C);
```

```
% Display the results
disp('Matrix C:');
disp(C);
disp('Lower Triangular Matrix L:');
disp(L);
disp('Upper Triangular Matrix U:');
disp(U);
```

Output:

Matrix C:

```
4    3    2    1
3    2    1    4
2    1    4    3
1    4    3    2
```

Lower Triangular Matrix L:

```
1    0    0    0
0.7500    1    0    0
0.5000    0.6000    1    0
```

```

0.2500 -1.2000 0.2118 1
Upper Triangular Matrix U:
    4    3    2    1
0 -0.2500 -0.5000 3.2500
    0        0 3.6000 1.4000
    0        0        0 -2.0471

```

6. Exercises on Linear Transformation

6.1. Characteristic Equation

In the context of linear systems or differential equations, the characteristic equation typically refers to an equation that involves the eigenvalues of a matrix or the roots of a differential equation.

Matrix Form:

- For a matrix A , the characteristic equation is given by $\det(A - \lambda I) = 0$, where λ are the eigen values of A .

MATLAB CODE:

1. MATLAB CODE:

```

% Define a matrix A
A = [1 2; 3 4];
% Calculate eigenvalues using built-in function eig()
eigenvalues = eig(A);
disp('Eigenvalues of matrix A:');
disp(eigenvalues);
% Alternatively, you can solve the characteristic equation manually %
characteristic equation is  $\det(A - \lambda * I) = 0$ 
% For our matrix A = [1 2; 3 4],
it becomes  $(1-\lambda)*(4-\lambda) - 6 = 0$ 
% Define coefficients of the characteristic polynomial a = 1; b = -(A(1,1) +
A(2,2)); c = det(A);
% Solve the quadratic characteristic equation  $ax^2 + bx + c = 0$  lambda = roots([a
b c]); disp('Eigenvalues calculated using characteristic equation:');
disp(lambda);

```

Try

- Instead of temperature apply heat flux on the sides of square and find the temperature distribution.
- Introduce a hole at the center of square and find the temperature distribution

6.2. Eigen Values

Given a square matrix A , an eigenvalue λ and its corresponding eigenvector \mathbf{v} satisfy the equation: $A\mathbf{v} = \lambda\mathbf{v}$

Hints

1.
Preferences: Structural Preprocessor:

```
% Example matrix
A = [4 -2 1; -2 4 -2; 1 -2 3];

% Compute eigenvalues
eigenvalues = eig(A);

% Display the eigenvalues
disp('Eigenvalues of A:');
disp(eigenvalues); • Solve → Current LS → Ok
General Post Proc:
• Plot results → Temperature distribution
```

Try

1. Instead of temperature apply heat flux on the sides of square and find the temperature distribution.
2. Introduce a hole at the center of square and find the temperature distribution

6.3. Eigen Vectors

Given a square matrix A , an eigenvalue λ and its corresponding eigenvector \mathbf{v} satisfy the equation:

$$A\mathbf{v} = \lambda\mathbf{v}$$

This equation states that when matrix A is applied to eigenvector \mathbf{v} , the result is a scaled version of \mathbf{v} by λ . Eigenvectors are thus directions in the vector space that are only scaled by the matrix A .

Hints

```
1. Preferences: Structural Preprocessor:
% Visualize eigenvectors
figure;
hold on;
plot([0 v(1,1)], [0 v(2,1)], 'b', 'LineWidth', 2); % Eigenvector 1
plot([0 v(1,2)], [0 v(2,2)], 'r', 'LineWidth', 2); % Eigenvector 2
axis equal;
legend('Eigenvector 1', 'Eigenvector 2');
title('Eigenvectors of Matrix A');
xlabel('x');
ylabel('y');
grid on; • Solve → Current LS → Ok
```

General Post Proc:

- Plot results → Temperature distribution

Try

1. Instead of temperature apply heat flux on the sides of square and find the temperature distribution.
2. Introduce a hole at the center of square and find the temperature distribution

7. Exercises on Differentiation and Integration

7.1 Higher order differential equations

A higher order differential equation generally takes the form:

$$F(x, y, \frac{dy}{dx}, \frac{d^2y}{dx^2}, \dots, \frac{d^ny}{dx^n}) = 0$$

where $y = y(x)$ is the unknown function, x is the independent variable, and n denotes the order of the highest derivative present.

Hints

1. MATLAB CODE:

```
% Define the differential equation function
function dydx = myODE(x, y)
dydx = zeros(2,1);
dydx(1) = y(2); % This is dy/dx
dydx(2) = -y(1); % This is d^2y/dx^2 + y
end
% Define the range of x values and initial conditions
xspan = [0 10];
y0 = [0 1]; % y(0) = 0, dy/dx(0) = 1
% Solve the differential equation
[x, y] = ode45(@myODE, xspan, y0);
% Plot the solution
plot(x, y(:,1)); % y(:,1) contains the values of y(x)
xlabel('x');
ylabel('y(x)');
```

2. Solution:

- Analysis Type → New analysis → Transient → Ok
- Parameters → functions → define/edit → type in result
- Select file → file name = transient → desktop → save
- Parameters → functions → read from file → open transient → give table parameter name cantilever
- Select loads → define loads → apply → structural → displacement → on keypoints → all DOF → keypoint 1 → ok

Try

1. Use a cantilever beam with same length as stated in above problem but with circular cross section and find its nonlinear behavior.
2. Use the same cantilever beam subjected to torque and find its nonlinear behavior.

7.2 Double Integrals

A double integral is an integral where the integrand is a function of two variables, typically denoted as $f(x,y)$, over a region in the xy -plane. The notation for a double integral over a region R is:

$$\iint_R f(x,y) \, dx \, dy$$

Hints

1. Preferences: Structural Preprocessor:

```
% Define the function to be integrated
fun = @(x, y) x^2 + y;

% Define the limits of integration
xmin = 0;
xmax = 1;
ymin = 0;
ymax = 2;

% Compute the double integral
Q = integral2(fun, xmin, xmax, ymin, ymax);

disp(['The value of the double integral is: ', num2str(Q)]);
Solution:
```

- Analysis Type → New analysis → Transient → Ok
- Parameters → functions → define/edit → type in result
- Select file → file name = transient → desktop → save
- Parameters → functions → read from file → open transient → give table parameter name cantilever
- Select loads → define loads → apply → structural → displacement → on keypoints → all DOF → keypoint 1 → ok

Try

1. Use a cantilever beam with same length as stated in above problem but with circular cross section and find its nonlinear behavior.
2. Use the same cantilever beam subjected to torque and find its nonlinear behavior.

7.3 Triple Integrals

The triple integral of a function $f(x, y, z)$ over a region D in space is denoted as:

$$\iiint_D f(x, y, z) \, dV$$

where dV represents the differential volume element. The region D is typically described using appropriate bounds for x , y , and z .

Hints

1. Preferences: Structural Preprocessor:

```
% Define the function to be integrated
f = @(x, y, z) x.^2 + y.^2 + z.^2;
% Define the limits for the variables
x_min = 0; x_max = 1;
y_min = 0; y_max = 2;
z_min = 0; z_max = 3;
% Evaluate the triple integral using 'integral3' function
integral_result = integral3(f, x_min, x_max, y_min, y_max, z_min, z_max);
disp(['The value of the triple integral is: ', num2str(integral_result)]);
```

Solution:

- Analysis Type → New analysis → Transient → Ok
- Parameters → functions → define/edit → type in result
- Select file → file name = transient → desktop → save
- Parameters → functions → read from file → open transient → give table parameter name cantilever
- Select loads → define loads → apply → structural → displacement → on keypoints → all DOF → keypoint 1 → ok

Try

1. Use a cantilever beam with same length as stated in above problem but with circular cross section and find its nonlinear behavior.
2. Use the same cantilever beam subjected to torque and find its nonlinear behavior.

8. Exercises on Numerical Differentiation and Integration

8.1 Trapezoidal rule

Numerical integration involves approximating the definite integral of a function over a specified interval. Popular methods include the Trapezoidal rule and Simpson's rule.

The Trapezoidal rule approximates the integral by approximating the area under the curve with trapezoids.

Hints

```
function I = trapezoidal_rule(f, a, b, n)
h = (b - a) / n;
x = a:h:b;
y = f(x);
I = h * (sum(y) - (y(1) + y(end)) / 2);
End
% Example usage:
% Define your function:
f = @(x) sin(x);
a = 0;
b = pi;
n = 100;
integral_approx = trapezoidal_rule(f, a, b, n);
```

- Element → Add → Beam 3
- Material Properties → Material Models → Structural → Linear → Elastic → Isotropic → Ex = 2e11 → PRXY = 0.33 → Density = 7850kg/(m³) → Ok
- Modeling → Create → Key points → Inactive CS → (0,0,0);(100,0,0) → Ok → Lines → Areas → Ok
- Meshing → Size Controls → Manual Sizing → Lines → Picked Lines → No. of elements = 20 → Ok → Mesh → Lines → Ok
- Select parameters → select functions define

1. Solution:

- Analysis Type → New analysis → Transient → Ok
- Parameters → functions → define/edit → type in result
- Select file → file name = transient → desktop → save
- Parameters → functions → read from file → open transient → give table parameter name cantilever
- Select loads → define loads → apply → structural → displacement → on keypoints → all DOF → keypoint 1 → ok

Try

1. Use a cantilever beam with same length as stated in above problem but with circular cross section and find its nonlinear behavior.
2. Use the same cantilever beam subjected to torque and find its nonlinear behavior.

8.2 Euler method

Given a first-order differential equation $\frac{dy}{dx} = f(x, y)$ with initial condition $y(x_0) = y_0$, the Euler method proceeds as follows:

1. **Discretize the Interval:** Choose a step size h .
2. **Iterate:** Use the formula $y_{n+1} = y_n + h \cdot f(x_n, y_n)$, where y_n is the approximation of y at x_n .

Hints

1. Preferences: Structural Preprocessor:
2. % Euler Method Example

```

3. % Solving dy/dx = x + y, y(0) = 1
4. % Define the function f(x, y)
5. f = @(x, y) x + y;
6. % Initial condition
7. x0 = 0;
8. y0 = 1;
9. % Step size
10. h = 0.1;
11. % Number of steps
12. N = 10;
13. % Arrays to store results
14. x = zeros(1, N+1);
15. y = zeros(1, N+1);
16. % Initial values
17. x(1) = x0;
18. y(1) = y0;
19. % Euler method iteration
20. for n = 1:N
21.     x(n+1) = x(n) + h;
22.     y(n+1) = y(n) + h * f(x(n), y(n));
23. end
24. % Plotting the results
25. plot(x, y, '-o');
26. xlabel('x');
27. ylabel('y');
28. title('Euler Method Solution: dy/dx = x + y');
29. grid on;Solution:

```

- Analysis Type → New analysis → Transient → Ok
- Parameters → functions → define/edit → type in result
- Select file → file name = transient → desktop → save
- Parameters → functions → read from file → open transient → give table parameter name cantilever
- Select loads → define loads → apply → structural → displacement → on keypoints → all DOF → keypoint 1 → ok

Try

1. Use a cantilever beam with same length as stated in above problem but with circular cross section and find its nonlinear behavior.
2. Use the same cantilever beam subjected to torque and find its nonlinear behavior.

8.3 RungeKutta method

The Runge-Kutta methods are a family of numerical techniques used for solving ordinary differential equations (ODEs). The most commonly used variant is the fourth-order Runge-Kutta method (RK4), which is known for its balance between accuracy and computational efficiency. Here's a brief outline of RK4:

Hints

1. Preferences: Structural Preprocessor:

```
• function [x, y] = RK4(f, x0, y0, h, xmax)
    x = x0:h:xmax;
    y = zeros(size(x));
    y(1) = y0;

    for n = 1:length(x)-1
        k1 = h * f(x(n), y(n));
        k2 = h * f(x(n) + h/2, y(n) + k1/2);
        k3 = h * f(x(n) + h/2, y(n) + k2/2);
        k4 = h * f(x(n) + h, y(n) + k3);

        y(n+1) = y(n) + (k1 + 2*k2 + 2*k3 + k4) / 6;
    end
endSolution:
```

- Analysis Type → New analysis → Transient → Ok
- Parameters → functions → define/edit → type in result
- Select file → file name = transient → desktop → save
- Parameters → functions → read from file → open transient → give table parameter name cantilever
- Select loads → define loads → apply → structural → displacement → on keypoints → all DOF → keypoint 1 → ok

Try

1. Use a cantilever beam with same length as stated in above problem but with circular cross section and find its nonlinear behavior.
2. Use the same cantilever beam subjected to torque and find its nonlinear behavior.

9. Exercises on 3D Plotting

9.1 Line plotting

A cantilever beam with length of 550cm and cross section area of 100 mm² subjected to nonlinear load. Find the deflection and nonlinear behavior.

Hints

1. Preferences: Structural Preprocessor:

- Element → Add → Beam 3
- Material Properties → Material Models → Structural → Linear → Elastic → Isotropic → $E = 2e11$ → $\nu = 0.33$ → Density = $7850 \text{ kg}/(\text{m}^3)$ → Ok
- Modeling → Create → Key points → Inactive CS → $(0,0,0);(100,0,0)$ → Ok → Lines → Areas → Ok
- Meshing → Size Controls → Manual Sizing → Lines → Picked Lines → No. of elements = 20 → Ok → Mesh → Lines → Ok
- Select parameters → select functions define

2. Solution:

- Analysis Type → New analysis → Transient → Ok
- Parameters → functions → define/edit → type in result
- Select file → file name = transient → desktop → save
- Parameters → functions → read from file → open transient → give table parameter name cantilever
- Select loads → define loads → apply → structural → displacement → on keypoints → all DOF → keypoint 1 → ok

Try

1. Use a cantilever beam with same length as stated in above problem but with circular cross section and find its nonlinear behavior.
2. Use the same cantilever beam subjected to torque and find its nonlinear behavior.

9.2 Surface plotting

A cantilever beam with length of 550cm and cross section area of 100 mm^2 subjected to nonlinear load. Find the deflection and nonlinear behavior.

Hints

1. Preferences: Structural Preprocessor:

- Element → Add → Beam 3
- Material Properties → Material Models → Structural → Linear → Elastic → Isotropic → $E = 2e11$ → $\nu = 0.33$ → Density = $7850 \text{ kg}/(\text{m}^3)$ → Ok
- Modeling → Create → Key points → Inactive CS → $(0,0,0);(100,0,0)$ → Ok → Lines → Areas → Ok
- Meshing → Size Controls → Manual Sizing → Lines → Picked Lines → No. of elements = 20 → Ok → Mesh → Lines → Ok
- Select parameters → select functions define

2. Solution:

- Analysis Type → New analysis → Transient → Ok
- Parameters → functions → define/edit → type in result
- Select file → file name = transient → desktop → save
- Parameters → functions → read from file → open transient → give table parameter name cantilever
- Select loads → define loads → apply → structural → displacement → on keypoints → all DOF → keypoint 1 → ok

Try

1. Use a cantilever beam with same length as stated in above problem but with circular cross section and find its nonlinear behavior.

2. Use the same cantilever beam subjected to torque and find its nonlinear behavior.

9.3 Volume plotting

A cantilever beam with length of 550cm and cross section area of 100 mm² subjected to nonlinear load.. Find the deflection and nonlinear behavior.

Hints

1. Preferences: Structural Preprocessor:

- Element → Add → Beam 3
- Material Properties → Material Models → Structural → Linear → Elastic → Isotropic → $E_x = 2e11$ → $\nu_{PRXY} = 0.33$ → Density = 7850kg/(m³) → Ok
- Modeling → Create → Key points → Inactive CS → (0,0,0);(100,0,0) → Ok → Lines → Areas → Ok
- Meshing → Size Controls → Manual Sizing → Lines → Picked Lines → No. of elements = 20 → Ok → Mesh → Lines → Ok
- Select parameters → select functions define

2. Solution:

- Analysis Type → New analysis → Transient → Ok
- Parameters → functions → define/edit → type in result
- Select file → file name = transient → desktop → save
- Parameters → functions → read from file → open transient → give table parameter name cantilever
- Select loads → define loads → apply → structural → displacement → on keypoints → all DOF → keypoint 1 → ok

Try

1. Use a cantilever beam with same length as stated in above problem but with circular cross section and find its nonlinear behavior.
2. Use the same cantilever beam subjected to torque and find its nonlinear behavior.

10. Exercises on Deflection of Simply Supported Beam

10.1 Calculating vertical displacement with point load

A cantilever beam made of mild steel as following specifications $L=150$ cm, $b=10$ cm, $h=10$ cm shown in Fig 8.1.is subjected to a periodic force, which is mathematically represented below. The amplitude of the force is 1000 N. $P = 1000k \sin\left(\frac{\pi t}{4}\right)$. Find the deflection of beam.

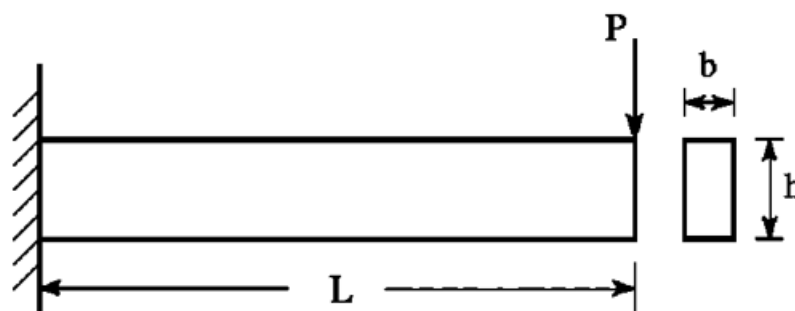


Fig 8.1. Cantilever beam with periodic force

Hints

1. Preferences: Structural Preprocessor:

- Element → Add → Beam 3
- Material Properties → Material Models → Structural → Linear → Elastic → Isotropic → $E_x = 2e11$ → $\nu_{xy} = 0.33$ → Density = $7850 \text{ kg}/(\text{m}^3)$ → Ok
- Modeling → Create → Key points → Inactive CS → $(0,0,0);(100,0,0)$ → Ok → Lines → Areas → Ok
- Meshing → Size Controls → Manual Sizing → Lines → Picked Lines → No. of elements = 20 → Ok → Mesh → Lines → Ok
- Select parameters → select functions define

3. Solution:

- Analysis Type → New analysis → → Ok
- Parameters → functions → define/edit → type in result $1000k \sin(\pi t/4)$
- Select file → file name = transient → desktop → save
- Parameters → functions → read from file → open transient → give table parameter name cantilever
- Select loads → define loads → apply → structural → displacement → on keypoints → all DOF → keypoint 1 → ok
- Solve → Current LS → Ok

Try

1. Repeat the above analysis for simply supported beam.
2. Change the magnitude of periodic load and repeat the same analysis.

10.2 Calculating vertical displacement with uniformly distributed load.

A cantilever beam made of mild steel as following specifications $L=150 \text{ cm}$, $b=10 \text{ cm}$, $h=10 \text{ cm}$ shown in Fig 8.1. is subjected to a periodic force, which is mathematically represented below. The amplitude of the force is 1000 N . $P = 1000k \sin(\frac{\pi t}{4})$. Find the deflection of beam.

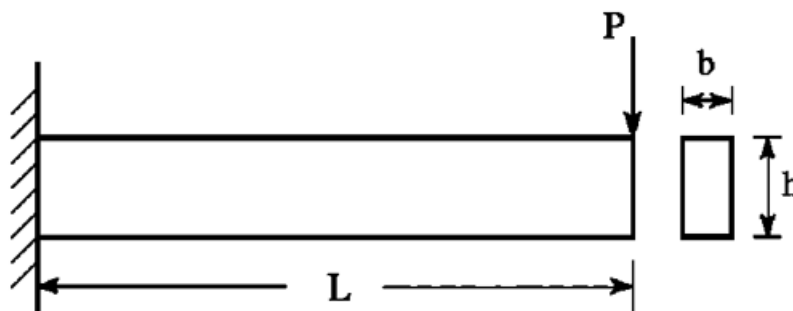


Fig 8.1. Cantilever beam with periodic force

Hints

1. Preferences: Structural Preprocessor:

- Element → Add → Beam 3
- Material Properties → Material Models → Structural → Linear → Elastic → Isotropic → $E_x = 2e11$ → $\nu_{xy} = 0.33$ → Density = $7850 \text{ kg}/(\text{m}^3)$ → Ok
- Modeling → Create → Key points → Inactive CS → $(0,0,0);(100,0,0)$ → Ok → Lines → Areas → Ok

- Meshing → Size Controls → Manual Sizing → Lines → Picked Lines → No. of elements = 20 → Ok → Mesh → Lines → Ok
- Select parameters → select functions define

2. Solution:

- Analysis Type → New analysis → → Ok
- Parameters → functions → define/edit → type in result $1000k \sin(\pi t/4)$
- Select file → file name = transient → desktop → save
- Parameters → functions → read from file → open transient → give table parameter name cantilever
- Select loads → define loads → apply → structural → displacement → on keypoints → all DOF → keypoint 1 → ok
- Solve → Current LS → Ok

Try

1. Repeat the above analysis for simply supported beam.
2. Change the magnitude of periodic load and repeat the same analysis.

10.3 Calculating vertical displacement with uniformly varying load

A cantilever beam made of mild steel as following specifications $L=150$ cm, $b=10$ cm, $h=10$ cm shown in Fig 8.1. is subjected to a periodic force, which is mathematically represented below. The amplitude of the force is 1000 N. $P = 1000k \sin(\frac{\pi t}{4})$. Find the deflection of beam.

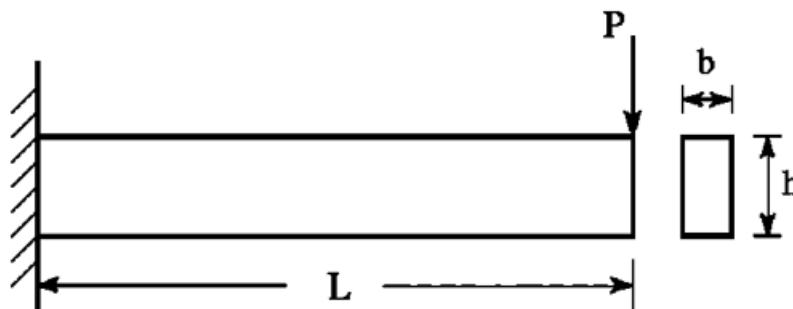


Fig 8.1. Cantilever beam with periodic force

Hints

1. Preferences: Structural Preprocessor:

- Element → Add → Beam 3
- Material Properties → Material Models → Structural → Linear → Elastic → Isotropic → Ex = $2e11$ → PRXY = 0.33 → Density = $7850\text{kg}/(\text{m}^3)$ → Ok
- Modeling → Create → Key points → Inactive CS → $(0,0,0);(100,0,0)$ → Ok → Lines → Areas → Ok
- Meshing → Size Controls → Manual Sizing → Lines → Picked Lines → No. of elements = 20 → Ok → Mesh → Lines → Ok
- Select parameters → select functions define

2. Solution:

- Analysis Type → New analysis → → Ok
- Parameters → functions → define/edit → type in result $1000k \sin(\pi t/4)$

- Select file→ file name= transient→ desktop→ save
- Parameters→ functions→ read from file→ open transient→ give table parameter name cantilever
- Select loads→ define loads→ apply→ structural→ displacement→ on keypoints→ all DOF→ keypoint 1→ ok
- Solve → Current LS → Ok

Try

1. Repeat the above analysis for simply supported beam.
2. Change the magnitude of periodic load and repeat the same analysis.

11. Exercises on Deflection of Cantilever Beam

11.1 Calculating vertical displacement with point load

A cantilever beam made of mild steel as following specifications $L=150$ cm, $b=10$ cm, $h=10$ cm shown in Fig 8.1.is subjected to a periodic force, which is mathematically represented below. The amplitude of the force is 1000 N. $P = 1000k \sin\left(\frac{\pi t}{4}\right)$. Find the deflection of beam.

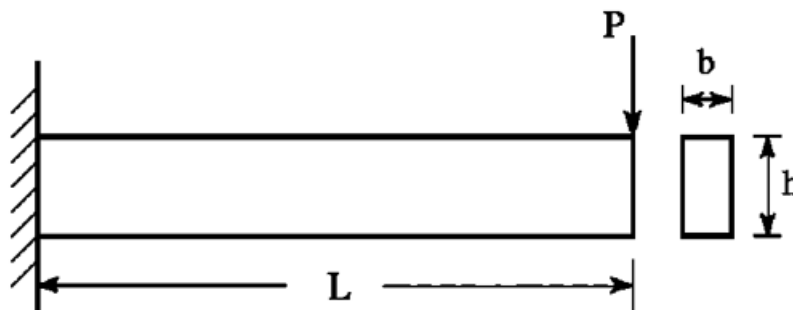


Fig 8.1. Cantilever beam with periodic force

Hints

1. Preferences: Structural Preprocessor:

- Element → Add → Beam 3
- Material Properties → Material Models → Structural → Linear → Elastic → Isotropic → $E_x = 2e11$ → $\nu_{xy} = 0.33$ → Density = $7850\text{kg}/(\text{m}^3)$ → Ok
- Modeling → Create → Key points → Inactive CS → $(0,0,0);(100,0,0)$ → Ok → Lines → Areas → Ok
- Meshing → Size Controls → Manual Sizing → Lines → Picked Lines → No. of elements = 20 → Ok → Mesh → Lines → Ok
- Select parameters → select functions define

2. Solution:

- Analysis Type → New analysis → → Ok
- Parameters → functions → define/edit → type in result $1000k \sin\left(\frac{\pi t}{4}\right)$
- Select file→ file name= transient→ desktop→ save
- Parameters→ functions→ read from file→ open transient→ give table parameter name cantilever
- Select loads→ define loads→ apply→ structural→ displacement→ on keypoints→ all DOF→ keypoint 1→ ok
- Solve → Current LS → Ok

Try

1. Repeat the above analysis for simply supported beam.
2. Change the magnitude of periodic load and repeat the same analysis.

11.2 Calculating vertical displacement with uniformly distributed load.

A cantilever beam made of mild steel as following specifications $L=150$ cm, $b=10$ cm, $h=10$ cm shown in Fig 8.1.is subjected to a periodic force, which is mathematically represented below. The amplitude of the force is 1000 N. $P = 1000k \sin(\frac{\pi t}{4})$. Find the deflection of beam.

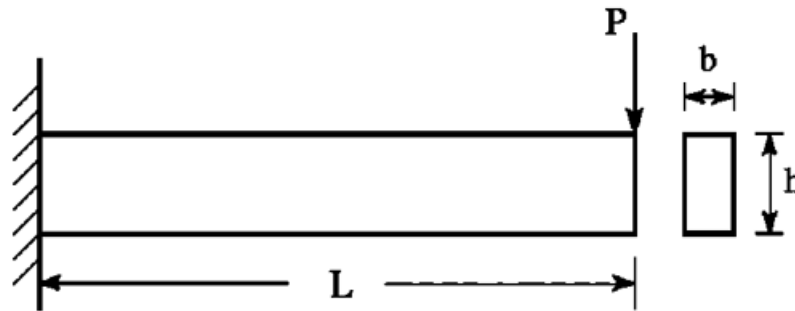


Fig 8.1. Cantilever beam with periodic force

Hints

1. Preferences: Structural Preprocessor:

- Element → Add → Beam 3
- Material Properties → Material Models → Structural → Linear → Elastic → Isotropic → $E_x = 2e11$ → $\nu_{xy} = 0.33$ → Density = $7850 \text{ kg}/(\text{m}^3)$ → Ok
- Modeling → Create → Key points → Inactive CS → $(0,0,0);(100,0,0)$ → Ok → Lines → Areas → Ok
- Meshing → Size Controls → Manual Sizing → Lines → Picked Lines → No. of elements = 20 → Ok → Mesh → Lines → Ok
- Select parameters → select functions define

2. Solution:

- Analysis Type → New analysis → → Ok
- Parameters → functions → define/edit → type in result $1000k \sin(\pi t/4)$
- Select file → file name = transient → desktop → save
- Parameters → functions → read from file → open transient → give table parameter name cantilever
- Select loads → define loads → apply → structural → displacement → on keypoints → all DOF → keypoint 1 → ok
- Solve → Current LS → Ok

Try

1. Repeat the above analysis for simply supported beam.
2. Change the magnitude of periodic load and repeat the same analysis.

11.3 Calculating vertical displacement with uniformly varying load

A cantilever beam made of mild steel as following specifications $L=150$ cm, $b=10$ cm, $h=10$ cm shown in Fig 8.1.is subjected to a periodic force, which is mathematically represented below. The amplitude of the force is 1000 N. $P = 1000k \sin(\frac{\pi t}{4})$. Find the deflection of beam.

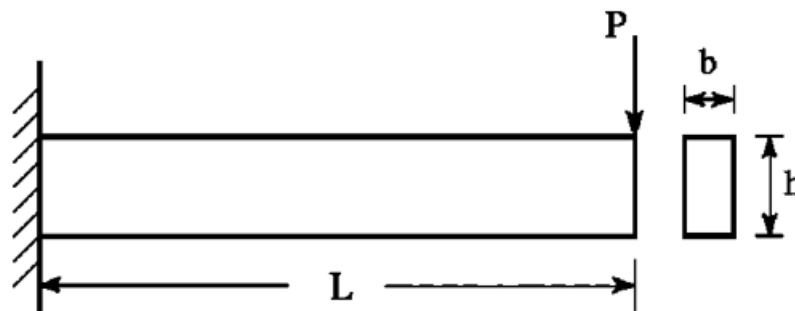


Fig 8.1. Cantilever beam with periodic force

Hints

1. Preferences: Structural Preprocessor:

- Element → Add → Beam 3
- Material Properties → Material Models → Structural → Linear → Elastic → Isotropic → $E_x = 2e11$ → $\nu_{PRXY} = 0.33$ → Density = $7850 \text{ kg}/(\text{m}^3)$ → Ok
- Modeling → Create → Key points → Inactive CS → $(0,0,0);(100,0,0)$ → Ok → Lines → Areas → Ok
- Meshing → Size Controls → Manual Sizing → Lines → Picked Lines → No. of elements = 20 → Ok → Mesh → Lines → Ok
- Select parameters → select functions define

2. Solution:

- Analysis Type → New analysis → → Ok
- Parameters → functions → define/edit → type in result $1000k \sin(\pi t/4)$
- Select file → file name= transient → desktop → save
- Parameters → functions → read from file → open transient → give table parameter name cantilever
- Select loads → define loads → apply → structural → displacement → on keypoints → all DOF → keypoint 1 → ok
- Solve → Current LS → Ok

Try

1. Repeat the above analysis for simply supported beam.
2. Change the magnitude of periodic load and repeat the same analysis.

12. Exercises on Formulation of Ideal and Real Gas Equations

12.1 Calculating the pressure, temperature, density for Earth's atmospheric conditions at different altitudes.

Perform static structural analysis of an aircraft rectangular wing of a span 100cm and the chord length of wing is 10 cm and cross-sectional area is 100 cm². Find the deflection and deformation of wing.

1. Preferences: Structural Preprocessor:

- Element → Add → wing • Real constants → Add Set1 → Cross section area = 100 cm^2 → $I_{zz} = 833.33 \text{ cm}^4$ → Height = 10 cm → Ok → Close
- Material Properties → Material Models → Structural → Linear → Elastic → Isotropic → $E_x = 2.5e7$ → $\nu_{xy} = 0.28$ → Ok
- Modeling → Create → Key points → Inactive CS → (0,0,0) ;(100,0,0) ;(150,0,0) → Ok → Lines → Straight Lines → Ok
- Meshing → Size Controls → Manual Sizing → Lines → Picked Lines → No. of elements = 20 → Ok → Mesh → Lines → Ok

2. Solution:

- Analysis Type → New analysis → Static → Ok
- Define Loads → Apply → Structural → Displacement → On Key Points → Ok → All DOF Constrained → Ok → Force/Moments → On Key Points → $F_y = -10000$ → Ok
- Solve → Current LS → Ok

3.General Post Proc:

- Plot results → Deformed Shapes → Deformed + Un deformed
- Contour Plots → Nodal Solutions → DOF Solution → Y- Component Displacement Ok

Try

1. Repeat the above analysis for a tapered wing.
2. Repeat the above analysis for tapered wing with uniformly varying load

12.2 Calculating the pressure, temperature, density for other planets at different altitudes.

Perform model analysis of an aircraft rectangular wing of a span 100 cm and the chord length of wing is 10cm and cross-sectional area is 100 cm^2 . Calculate the natural frequency of wing.

Hints

1. Preferences: Structural Preprocessor:

- Element → Add → wing
- Real constants → Add Set1 → Cross section area = 100 cm^2 → $I_{zz} = 833.33 \text{ cm}^4$ → Height = 10 cm → Ok → Close
- Material Properties → Material Models → Structural → Linear → Elastic → Isotropic → $E_x = 2.5e7$ → $\nu_{xy} = 0.28$ → Density = $7800 \text{ kg}/(\text{m}^3)$ → Ok
- Modeling → Create → Key points → Inactive CS → (0,0,0) ;(100,0,0) ;(150,0,0) → Ok → Lines → Straight Lines → Ok
- Meshing → Size Controls → Manual Sizing → Lines → Picked Lines → No. of elements = 10 → Ok → Mesh → Lines → Ok 6.4.2

2. Solution:

- Analysis Type → New analysis → Modal → Ok
- Analysis Option → Block Lancos → No. of Modes to extract = 5 → No. of Modes to expand = 5 → Ok
- Solve → Current LS → Ok

Try

1. Repeat the above analysis for a tapered wing.
2. Repeat the above analysis for tapered wing with uniformly varying load

13. Exercises on Shear Force and Bending Moment Diagrams - Cantilever Beam

13.1 Calculating shear force and bending moment for point load.

Calculate the deformation of the aluminum fuselage section under the application of internal load of 100000 Pa. The radius of fuselage is 0.15m and thickness is 2 mm.

1. Preferences: Structural Preprocessor:

- Element type → Add / edit/Delete → Add → Solid - 10 node 92 → Apply Add → Beam 2 Node 188 → Apply → Add → Shell → Elastic 4 node 63 Real Constants → Add → Select shell → give thickness (I) = 1 → ok → close.
- Material properties → material models → Structural → Linear → Elastic → Isotropic EX = 0.7e11; PRXY = 0.3; Density = 2700
- Pre-processor → modelling → Create → Areas → Circle → Annulus WP x = 0 ; WP y = 0; Rad - 1 = 2.5; Rad - 2 = 2.3 OK Pre-processor → Modelling → Create → Circle → Solid - WP x = 0; X = 2.25; Y = 0 Radius = 0.15 Apply WP x = 0; X = -2.25; Y = 0 Radius = 0.15 Apply WP x = 0; X = 0; Y = 2.25; Radius = 0.15 Apply WP x = 0; X = 0; Y = -2.25 Radius = 0.15 OK
- Pre-processor → Modelling → Operate → Booleans → Add → Areas - Pick all OK
- Preprocessor → Modelling → Operate → Extrude → Areas → By XYZ offset X= 0; Y=0; Z = 5
- Pre-processor → Meshing → Size controls → Manual Size → All Areas → give element edge length as 0.15 → ok
- Meshing → Size controls → Manual Size → All lines → give element edge length as → ok
- Meshing → Mesh → areas → free → select box type instead of single → select the total volume → ok

2. Solution:

- Loads → define loads → Apply → Structural → Displacement → On areas → select box type → select box (4 points at centre) → all DOF → ok Select → ALL DOF arrested Define loads → Apply → Structural → Pressure → on areas → select the internal surface of the fuselage and give value (100000) → ok
- Solve → Current LS → Ok
- General Post Proc:
- Plot results → Deformed Shapes → Deformed + Un deformed

Try

1. Repeat the above analysis for a monocoque fuselage.
2. Repeat the above analysis for a monocoque fuselage with both internal and external load.

13.2 Calculating shear force and bending moment for uniformly distributed load.

Calculate the natural frequency of the aluminum fuselage section under the application of internal load of 100000 Pa. The radius of fuselage is 0.15m and thickness is 2 mm.

1. Preferences: Structural Preprocessor:

- Element type → Add / edit/Delete → Add → Solid - 10 node 92→ Apply Add → Beam 2 Node 188 → Apply → Add → Shell →Elastic 4 node 63 Real Constants → Add → Select shell → give thickness (I) = 1→ ok → close.
- Material properties → material models → Structural → Linear → Elastic → Isotropic EX =0.7e11; PRXY = 0.3; Density = 2700
- Pre-processor → modelling → Create → Areas → Circle → Annulus WP x = 0 ; WP y = 0; Rad - 1 = 2.5; Rad - 2 = 2.3 OK Pre-processor → Modelling → Create → Circle → Solid -WP x = 0; X = 2.25; Y = 0 Radius = 0.15 Apply WP x = 0; X = -2.25; Y = 0 Radius = 0.15 Apply WP x = 0; X = 0; Y = 2.25; Radius = 0.15 Apply WP x = 0; X = 0; Y = -2.25 Radius = 0.15 OK
- Pre-processor → Modelling → Operate → Booleans → Add → Areas - Pick all OK
- Preprocessor → Modelling → Operate → Extrude → Areas → By XYZ offset X= 0; Y=0; Z = 5
- Pre-processor → Meshing → Size controls → Manual Size → All Areas → give element edge length as 0.15 → ok
- Meshing → Size controls → Manual Size → All lines → give element edge length as→ ok
- Meshing → Mesh → areas → free → select box type instead of single → select the total volume → ok

2. Solution:

- Analysis Type → New analysis → Modal → Ok
- Analysis Option → Block Lancos → No. of Modes to extract = 5 → No. of Modes to expand = 5 → Ok
- Solve → Current LS → Ok

Try

1. Repeat the above analysis for a monocoque fuselage.
2. Repeat the above analysis for a monocoque fuselage with both internal and external load.

13.3 Calculating shear force and bending moment for uniformly varying load.

Calculate the natural frequency of the aluminum fuselage section under the application of internal load of 100000 Pa. The radius of fuselage is 0.15m and thickness is 2 mm.

1. Preferences: Structural Preprocessor:

- Element type → Add / edit/Delete → Add → Solid - 10 node 92→ Apply Add → Beam 2 Node 188 → Apply → Add → Shell →Elastic 4 node 63 Real Constants → Add → Select shell → give thickness (I) = 1→ ok → close.
- Material properties → material models → Structural → Linear → Elastic → Isotropic EX =0.7e11; PRXY = 0.3; Density = 2700
- Pre-processor → modelling → Create → Areas → Circle → Annulus WP x = 0 ; WP y = 0; Rad - 1 = 2.5; Rad - 2 = 2.3 OK Pre-processor → Modelling → Create →

Circle → Solid -WP x = 0; X = 2.25; Y = 0 Radius = 0.15 Apply WP x = 0; X = -2.25; Y = 0 Radius = 0.15 Apply WP x = 0; X = 0; Y = 2.25; Radius = 0.15 Apply WP x = 0; X = 0; Y = -2.25 Radius = 0.15 OK

- Pre-processor → Modelling → Operate → Booleans → Add → Areas - Pick all OK
- Preprocessor → Modelling → Operate → Extrude → Areas → By XYZ offset X= 0; Y=0; Z = 5
- Pre-processor → Meshing → Size controls → Manual Size → All Areas → give element edge length as 0.15 → ok
- Meshing → Size controls → Manual Size → All lines → give element edge length as → ok
- Meshing → Mesh → areas → free → select box type instead of single → select the total volume → ok

2. Solution:

- Analysis Type → New analysis → Modal → Ok
- Analysis Option → Block Lancos → No. of Modes to extract = 5 → No. of Modes to expand = 5 → Ok
- Solve → Current LS → Ok

Try

1. Repeat the above analysis for a monocoque fuselage.
2. Repeat the above analysis for a monocoque fuselage with both internal and external load.

14.Exercises on Shear Force and Bending Moment Diagrams- Over Hanging Beam

14.1 Calculating shear force and bending moment for point load

A simple retractable landing gear subjected to a load of 10000 N. Find the deformation and stress developed in the landing gear.

1. Preferences: Structural Preprocessor:

- Preferences → Structural → H-Method → OK
- Preprocessor → Element Type → Add → Add → Select Link → 2D spar 1 → Apply
- Preprocessor → Element Type → Add → Add → Select Beam → 2 Node 188 → OK → Close
- RealConstants → Add → Add → Select Type Link 1 → Click OK
- Enter the cross-sectional area =1 → OK → Close
- Material Properties → Material Models → Structural → Linear → Elastic → Isotropic Enter
the Young's Modulus (EXY) = $3e7$
Poisson's Ratio (PRXY) = 0.3
- Sections → Beam → Common Sections → Subtype → Select Solid Circle R=0.5 N=20
T=0, Mesh view
- Preprocessor → Modeling → Create → Key points → In Active CS
- Create the key points according to the table

KP no	X	Y	Z
1	0	0	0
2	-12	0	0
3	12	0	0
4	0	-12	0
5	0	-12	0
6	0	12	0

- Modeling → Create → Lines → Lines → Straight Lines →
- Join the key points according to table

Line no	Join
1	1&4
2	4&5
3	5&6
4	2&5
5	3&4

- Preprocessor → Meshing → Mesh Attributes → All lines → Select element type Beam 188, Ok
- Meshing → Mesh tool→ set →Global 1Link1→Ok
- Lines→set→3&4 line click→2&5 line click→ok No of divisions 1→ok
- Mesh Tool→Mesh→Mesh only strut→ok Meshing → Mesh tool→ set →Global 2 Beam 188→Ok Lines→set→1&4 line click→4&5 line click→5&6 line click→ok Element egde length→1→ok
- Mesh Tool→Mesh→Mesh only Vertical line→ok Main menu→ plot Cntrls → Style → Size and Shape Click in the box against Display Element Type

2. Solution:

- Define Loads → Apply → Structural → Displacement → On key Points → Select keypoints 2 & 3→select UX,UY,UZ,ROTX,ROTY→Ok Select keypoints 2 & 3→select UX,UZ→Ok
- Modeling →Create;Nodes→Rotate nodes CS;Byangles;click 6th keypoint THXY →60→ok •Loads;Apply→Structural;Force/Moment→ click On nodes 28/Key point 6→Force/Moment value →10000
- Solution → Solve→ General Post proc→ List results→ Rection solution→ Plot results→ Defromed shape

Try

1. Repeat the above analysis for a landing gear subjected to compression load.

14.2 Calculating shear force and bending moment for uniformly distributed load.

A simple retractable landing gear subjected to a load of 10000 N. Find the deformation and stress developed in the landing gear.

1. Preferences: Structural Preprocessor:

- Preferences → Structural → H-Method → OK
- Preprocessor → Element Type → Add → Add → Select Link → 2D spar 1 → Apply
- Preprocessor → Element Type → Add → Add → Select Beam → 2 Node 188 → OK → Close
- RealConstants → Add → Add → Select Type Link 1 → Click OK

- Enter the cross sectional area =1 → OK → Close
- Material Properties → Material Models → Structural → Linear → Elastic → Isotropic Enter
- the Young's Modulus (EXY) = 3e7
- Poisson's Ratio (PRXY) = 0.3
- Sections → Beam → Common Sections → Subtype → Select Solid Circle R=0.5 N=20 T=0, Mesh view
- Preprocessor → Modeling → Create → Key points → In Active CS
- Create the key points according to the table

KP no	X	Y	Z
1	0	0	0
2	-12	0	0
3	12	0	0
4	0	-12	0
5	0	-12	0
6	0	12	0

- Modeling → Create → Lines → Lines → Straight Lines →
- Join the key points according to table

Line no	Join
1	1&4
2	4&5
3	5&6
4	2&5
5	3&4

- Preprocessor → Meshing → Mesh Attributes → All lines → Select element type Beam 188, Ok
- Meshing → Mesh tool → set → Global 1 Link1 → Ok
- Lines → set → 3&4 line click → 2&5 line click → ok No of divisions 1 → ok
- Mesh Tool → Mesh → Mesh only strut → ok Meshing → Mesh tool → set → Global 2 Beam 188 → Ok Lines → set → 1&4 line click → 4&5 line click → 5&6 line click → ok Element egde length → 1 → ok
- Mesh Tool → Mesh → Mesh only Vertical line → ok Main menu → plot Cntrls → Style → Size and Shape Click in the box against Display Element Type

3. Solution:

- Analysis Type → New analysis → Modal → Ok
- Analysis Option → Block Lancos → No. of Modes to extract = 5 → No. of Modes to expand = 5 → Ok
- Solve → Current LS → Ok

Try

1. Repeat the above analysis for a landing gear subjected to compression load.

14.3. Calculating shear force and bending moment for uniformly varying load

A simple retractable landing gear subjected to a load of 10000 N. Find the deformation and stress developed in the landing gear.

1. Preferences: Structural Preprocessor:

- Preferences → Structural → H-Method → OK

- Preprocessor → Element Type → Add → Add → Select Link → 2D spar 1 → Apply
- Preprocessor → Element Type → Add → Add → Select Beam → 2 Node 188 → OK → Close
- RealConstants → Add → Add → Select Type Link 1 → Click OK
- Enter the cross sectional area =1 → OK → Close
- Material Properties → Material Models → Structural → Linear → Elastic → Isotropic Enter the Young's Modulus (EXY) = 3e7
Poisson's Ratio (PRXY) = 0.3
- Sections → Beam → Common Sections → Subtype → Select Solid Circle R=0.5 N=20 T=0, Mesh view
- Preprocessor → Modeling → Create → Key points → In Active CS
- Create the key points according to the table

KP no	X	Y	Z
1	0	0	0
2	-12	0	0
3	12	0	0
4	0	-12	0
5	0	-12	0
6	0	12	0

- Modeling → Create → Lines → Lines → Straight Lines →
- Join the key points according to table

Line no	Join
1	1&4
2	4&5
3	5&6
4	2&5
5	3&4

- Preprocessor → Meshing → Mesh Attributes → All lines → Select element type Beam 188, Ok
- Meshing → Mesh tool→ set →Global 1Link1→Ok
- Lines→set→3&4 line click→2&5 line click→ok No of divisions 1→ok
- Mesh Tool→Mesh→Mesh only strut→ok Meshing → Mesh tool→ set →Global 2 Beam 188→Ok Lines→set→1&4 line click→4&5 line click→5&6 line click→ok Element egde length→1→ok
- Mesh Tool→Mesh→Mesh only Vertical line→ok Main menu→ plot Cntrls → Style → Size and Shape Click in the box against Display Element Type

2. Solution:

- Analysis Type → New analysis → Modal → Ok
- Analysis Option → Block Lancos → No. of Modes to extract = 5 → No. of Modes to expand = 5 → Ok
- Solve → Current LS → Ok

Try

1. Repeat the above analysis for a landing gear subjected to compression load.

V. TEXT BOOKS:

1. Cleve Moler, "Numerical Computing with MATLAB", SIAM, Philadelphia, 2nd edition, 2008.

VI. REFERENCE BOOKS:

1. Dean G. Duffy, "Advanced Engineering Mathematics with MATLAB", CRC Press, Taylor & Francis Group, 6th edition, 2015.
2. Delores M. Etter, David C. Kuncicky, Holly Moore, "Introduction to MATLAB 7", Pearson Education Inc, 1st edition, 2009.

3. Rao. V. Dukkipati, “MATLAB for ME Engineers”, New Age Science, 1st edition, 2008.

VII. ELECTRONICS RESOURCES:

1. <http://www.tutorialspoint.com/matlab/>

VIII. MATERIALS ONLINE

1. Course template
2. Lab manual



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

MECHANICS OF SOLIDS AND FLUID DYNAMICS LABORATORY								
III Semester: AE								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AAED05	Core	L	T	P	C	CIA	SEE	Total
		-	-	2	1	40	60	100
Contact Classes: Nil	Tutorial Classes: Nil	Practical Classes: 45			Total Classes: 45			
Prerequisite: Mechanics of solids, Fluid Dynamics								

I. COURSE OVERVIEW:

The Mechanics of Solids and Fluid Dynamics Laboratory provides undergraduate students with a comprehensive understanding of both material strength and fluid behavior through hands-on experiments. In the Mechanics of Solids section, students perform experiments to measure material properties such as impact strength, tensile strength, compressive strength, hardness, and ductility. This lab is also utilized for project work, testing various materials like composites, ferrous and nonferrous alloys, and identifying suitable materials for aero structures based on their mechanical properties.

The Fluid Dynamics section explores fluid properties and involves experiments with incompressible flow. Students gain fundamental knowledge of basic measurements and devices used in fluid dynamics. This course introduces flow behavior, fluid forces, and analytical tools. It covers various flow measurement devices, pumps, and turbines, teaching students how to assess their performance. Hands-on experience includes investigating fluid statics principles, the kinematics and kinetics of fluid flow, and the operation of turbomachinery. This integrated laboratory ensures a thorough practical understanding of both solid mechanics and fluid dynamics, essential for applications in engineering and research.

II. COURSE OBJECTIVES:

The students will try to learn:

- I. The mechanics of materials and structural analysis through a series of experiments using appropriate codes and standards
- II. The behavior and failure modes using deflection of beams and columns for choosing the safety factor for engineering applications.
- III. Application of Bernoulli's theorem in measurement of rate of discharge in pumps.
- IV. The flow measurement and performance of pumps and turbines under various speeds.

III. COURSE OUTCOMES:

After successful completion of the course, students will be able to:

- CO1 Compare the hardness of ferrous and non-ferrous materials using hardness testing machines for identifying suitable industrial applications.
- CO2 Choose the regions of elasticity and plasticity, stress-strain relationships using universal testing machine for determining the safety factor.
- CO3 Summarize performance of a material or product using torsion tests, when undergoes rotational motion when in service.
- CO4 Demonstrate the validation of Bernoulli's theorem for incompressible, steady, continuous flow in order for regulating discharges in pipes.
- CO5 Illustrate the critical Reynolds number using Reynolds apparatus for transition of laminar flow into turbulent flow.
- CO6 Make use of the jet impact apparatus to investigate the reaction forces generated due to changes in momentum.

IV. COURSE CONTENT:

EXERCISES ON MECHANICS OF SOLIDS LABORATORY

1. Getting Started Exercises

1.1 Introduction to Laboratory

The Mechanics of Solids (MoS) Laboratory is equipped with advanced destructive testing machinery, enabling students to grasp fundamental concepts and apply them to practical problems. The laboratory conducts various tests as per ASTM and IS standards to estimate mechanical properties such as Young's Modulus, Shear Modulus, Hardness, Toughness, and Stiffness. Final-year students utilize this lab to test materials like composites, ferrous and nonferrous alloys, for designing components in aircraft and machine elements. The lab aims to:

- Familiarize students with lab equipment
- Inform students about the lab evaluation process
- Provide guidance on laboratory precautions
- Introduce sample preparation techniques

The Fluid Dynamics Laboratory is designed to explore fluid properties and conduct experiments with incompressible flow. This course provides fundamental knowledge of basic measurements and devices used in fluid dynamics. It introduces flow behaviour, fluid forces, and analytical tools. Students learn about various flow measurement devices, pumps, and turbines, and assess their performance. Hands-on experience covers principles of fluid statics, kinematics and kinetics of fluid flow, and turbo machinery operation. This comprehensive approach ensures students gain practical insights into fluid dynamics and its applications.

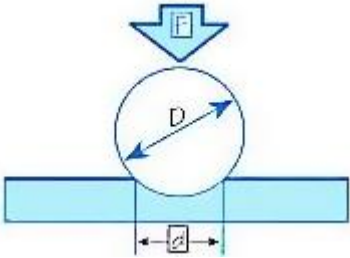
2. Brinell Hardness test

2.1. Introduction

Hardness is defined as a material's ability to resist permanent indentation (that is plastic deformation). Typically, the harder the material, the better it resists wear or deformation. The term hardness, thus, also refers to local surface stiffness of a material or its resistance to scratching, abrasion, or cutting.

2.2. Objectives

1. To determine the hardness number from Brinell hardness test.
2. To measure the ultimate tensile strength of the specimen from the Brinell hardness test.
3. The specimen prepared as per the given dimensions in fig.1

 <p>Fig.1: Brinell hardness test</p>	<p>Where D = Ball diameter d = impression diameter F = load HB = Brinell result</p>
---	--

2.3 Further Probing Experiments

1. Measure the hardness values for different conventional materials.
2. Compare the hardness values same material with different loads and comment on the results.

3. Tension test of Mild Steel

3.1. Introduction

The objective of this experiment is to evaluate the mechanical (tensile) properties of selected metallic materials using the tensile test method. These mechanical properties include modulus of elasticity, yield strength, ultimate tensile strength, failure strength, ductility, and strain to failure.

3.2. Procedure

1. Measure the original length and diameter of the specimen. The length may either be length of gauge section which is marked on the specimen with a preset punch or the total length of the specimen.
2. Insert the specimen into grips of the test machine and attach strain-measuring device to it.
3. Begin the load application and record load versus elongation data.
4. Take readings more frequently as yield point is approached.
5. Measure elongation values with the help of dividers and a ruler.
6. Continue the test till Fracture occurs.
7. By joining the two broken halves of the specimen together, measure the final length and diameter of specimen as shown in fig.2.

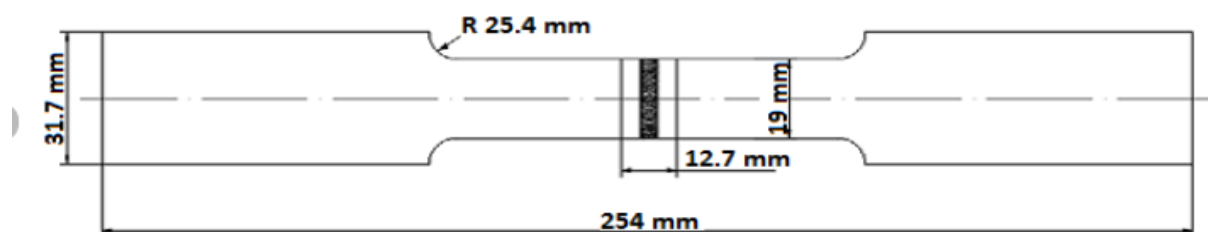


Fig.2: Tension test specimen preparation

3.3.Further Probing Experiments

1. Calculate the Young's Modulus for ductile materials.
2. Generate stress Vs strain diagram for different materials using servo driven Universal Testing Machine.

4. Torsion test of Mild Steel

4.1.Introduction

The stress resulting from torsion load can be determined by means of the torsion test. This test resembles the tension test in that a load deflection curve is also development (which is transformed to a shear-strain curve). In a torsion test, a solid or hollow cylindrical specimen is twisted and the resultant deformation, measured as the angle through which the bar is twisted. The test then consists of measuring the angle of twist, Φ (rad) at selected increments of torque, T (N.m). Expressing Φ as the angular deflection curve per unit gage length.

4.2.Procedure

1. Measure the original length and diameter of the specimen. The length may either be length of gauge section which is marked on the specimen with a preset punch or the total length of the specimen.
2. Insert the specimen into grips of the test machine and attach strain-measuring device to it.
3. Begin the load application and record load versus elongation data.
4. Take readings more frequently as yield point is approached.
5. Measure elongation values with the help of dividers and a ruler.
6. Continue the test till Fracture occurs.
7. By joining the two broken halves of the specimen together, measure the final length and diameter of specimen as shown in fig.3. (All dimensions are in mm).

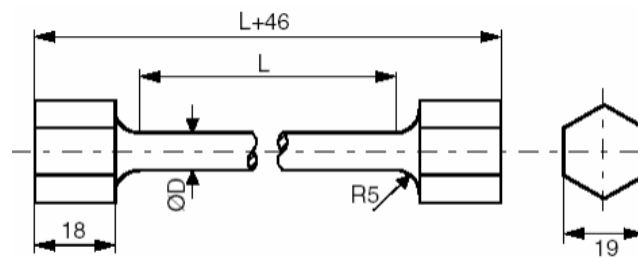


Fig.3 Torsion test specimen dimensions

4.3.Precautions

1. Wear tight overalls and shoe for safety.
2. If the strain measuring device is an extensometer it should be removed before necking begins.

3. Measure deflection on scale accurately and carefully.

4.4. Further Probing Experiments

1. Calculate the Young's Modulus for various ductile materials.
2. Generate stress Vs strain diagram for different materials using servo driven Universal Testing Machine.

5. Izod Impact test

5.1. Introduction

Impact test determines the amount of energy absorbed by a material during fracture. This absorbed energy is a measure of a given material's toughness and acts as a tool to study temperature-dependent brittle-ductile transition.

5.2. Procedure

1. With the striking hammer (pendulum) in safe test position, firmly hold the steel specimen in impact testing machines vice in such a way that the notch faces the hammer and is half inside and half above the top surface of the vice.
2. Bring the striking hammer to its top most striking position unless it is already there, and lock it at that position.
3. Bring indicator of the machine to zero, or follow the instructions of the operating manual supplied with the machine.
4. Release the hammer. It will fall due to gravity and break the specimen through its momentum, the total energy is not absorbed by the specimen. Then it continues to swing.
5. At its topmost height after breaking the specimen, the indicator stops moving, while the pendulum falls back. Note the indicator at that topmost final position as shown in fig.4

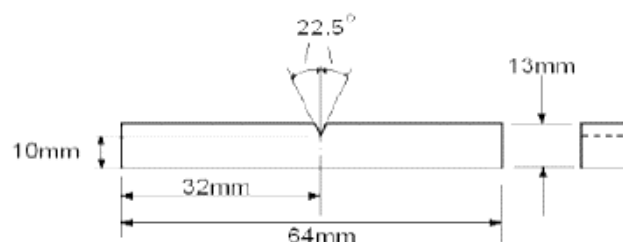


Fig.4: Izod impact test specimen dimensions

6. Charpy impact test

6.2 Introduction

Impact test determines the amount of energy absorbed by a material during fracture. This absorbed energy is a measure of a given material's toughness and acts as a tool to study temperature-dependent brittle-ductile transition.

6.3 Procedure

1. With the striking hammer (pendulum) in safe test position, firmly hold the steel specimen in impact testing machines vice in such a way that the notch faces the hammer and is half inside and half above the top surface of the vice.
2. Bring the striking hammer to its top most striking position unless it is already there, and lock it at that position.
3. Bring indicator of the machine to zero, or follow the instructions of the operating manual supplied with the machine as shown in figure.5 below.

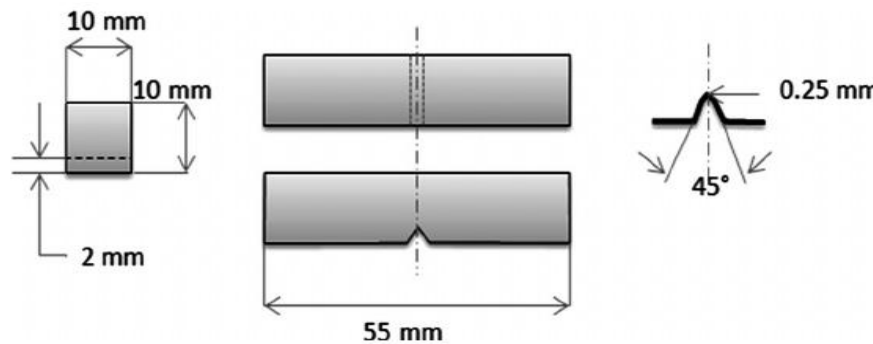


Fig.5. Charpy impact test specimen dimensions

6.4 Further Probing Experiments

1. Calculate the impact strength of unnotched specimens.
2. Determine the impact strength of U-Notched specimens.
3. Calculate the impact strength of a unnotched specimens.
4. Determine the impact strength of a U-Notched specimens.

7. Compression test

7.1 Short columns

Compression tests on short columns are used to determine a material's behaviour under applied crushing loads, and are typically conducted by applying compressive pressure using a universal testing machine. By using this experiment the load corresponding to the crushing stress, is called crushing load will be determined for the short columns.

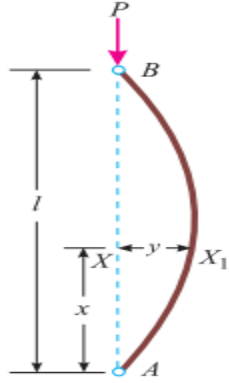
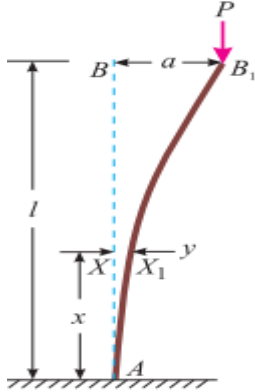
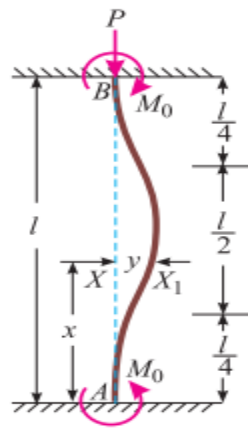
During the test, various properties of the material are calculated and plotted as a stress-strain diagram which is used to determine qualities such as elastic limit, proportional limit, yield point, yield strength and, for some materials, compressive strength

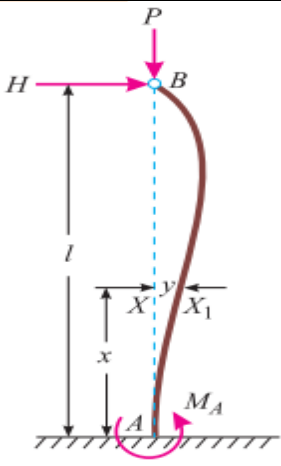
7.2 Long columns

Compression tests on long columns are used to determine a material's behaviour under applied buckling loads, and are typically conducted by applying compressive pressure using a universal testing machine. The column will fail due to buckling before the yield strength of the member is reached. Buckling occurs suddenly, and is characterized by large deflections perpendicular to the axis of the column. The stability of long columns also effects end conditions, which are 1. Both ends hinged, 2. Both ends fixed, 3. One end is fixed and the other hinged, and 4. One end is fixed and the other free.

During the test, various properties of the material are calculated and plotted as a stress-strain diagram which is used to determine qualities such as elastic limit, proportional limit, yield point, yield strength and, for some materials, compressive strength (Table 1).

Table 1.

S.No	Columns	Types of end conditions,
1	Columns with Both Ends Hinged	
2	Columns with One End Fixed and the Other Free	
3	Columns with Both Ends Fixed	

4	Columns with One End Fixed and the Other Hinged	
---	---	---

7.3 Further Probing Experiments

1. Calculate the compressive strength of a non ferrous metal cube.
2. Determine the compressive strength of a metallic columns
3. Calculate the compressive strength of a non ferrous metal cube.
4. Determine the compressive strength of a metallic columns

8. Spring test

8.1 Introduction

Spring test is used to determine the stiffness of helical spring. Stiffness is the ability of a material withstand load per unit deflection. The modulus of rigidity of a spring material varies as a function of chemical composition, cold working, and degree of aging,

8.2 Procedure

1. Measure the diameter of the wire of the spring by using the micrometre.
2. Measure the diameter of spring coils by using the Vernier calliper
3. Count the number of turns.
4. Insert the spring in the spring testing machine and load the spring by a suitable weight and note the corresponding axial deflection in tension or compression.
5. Increase the load and take the corresponding axial deflection readings.
6. Plot a curve between load and deflection. The shape of the curve gives the stiffness of the spring.

8.3 Further Probing Experiments

1. Measure the hardness values for different materials which needs unusual scales.
2. Compare the hardness values same material with different loads and comment on the results.

9. Deflection test for cantilever beam

9.1.1 Introduction

In Designing of beams two design criteria are important, one Strength which is resist to shear force and bending moment, and other is the stiffness, which is resistance to deflection under different types of loads. There are many methods to find out the slope and deflection at a section in a loaded beam, but 1. Double integration method and 2. Macaulay's methods are the important.

9.1.2 Procedure

In this experiment Macaulay's method is using out the slopes and deflection of beams with the following rules

- Always take origin on the extreme left of the beam.
- Take left clockwise moment as negative and left anticlockwise moment as positive.
- While calculating the slopes and deflections, it is convenient to use the values first in terms of kN and metres as shown in figure 6.

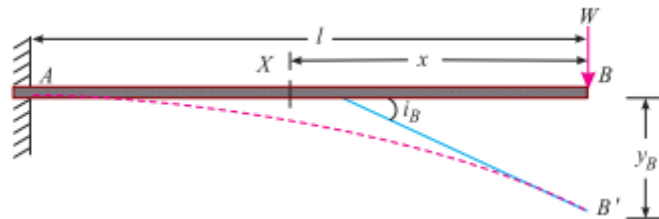


Fig. 6. Experiment Macaulay Method

9.2 Deflection test for simple supported beam

9.2.1 Introduction

In Designing of beams two design criteria are important, one Strength which is resist to shear force and bending moment, and other is the stiffness, which is resistance to deflection under different types of loads. There are many methods to find out the slope and deflection at a section in a loaded beam, but 1. Double integration method and 2. Macaulay's methods are the important.

9.2.2 Procedure

In this experiment Macaulay's method (Fig.7) is using out the slopes and deflection of beams with the following rules

- Always take origin on the extreme left of the beam.
- Take left clockwise moment as negative and left anticlockwise moment as positive.
- While calculating the slopes and deflections, it is convenient to use the values first in terms of kN and metres.

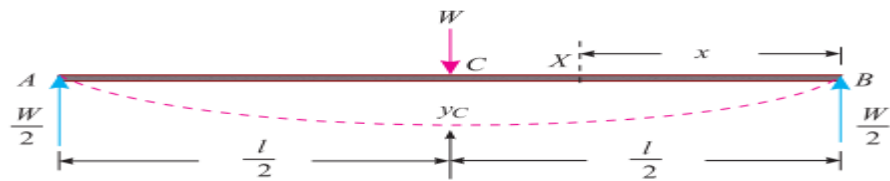


Fig.7. Demonstration of forces for Macaulay,s method

9.3 Further Probing Experiments

1. Measure the slope and deflection values for cantilever beam with different loads.
2. Repeat the same experiment, by changing the point of applications of loads.

10. Exercises on Verification of Bernoulli's theorem.

10.1 Bernoulli's theorem

Start the pump, adjust the flow, note down the piezometer readings and time (t), calculate the pressure head, velocity head, and datum head, and verify the Bernoulli's Theorem, using the experimental setup, shown in figure 8.

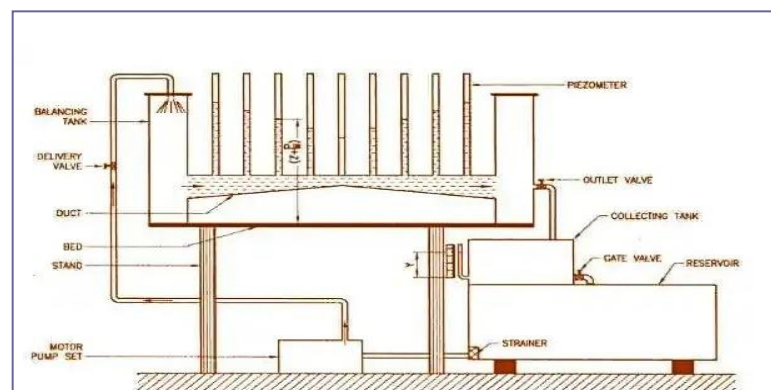


Fig.8. Bernoulli experiment setup

Try

1. Vary the mass flow rate and verify Bernoulli's theorem
2. Change the fluid type and verify Bernoulli's theorem

11. Exercises on Impact of Jets on Vanes

11.1 Jets on Vanes

1. Fix the given vane and add dead weight, note down the forces, note down the time, calculate the flow speed, discharge, and coefficient of impact vanes, shown in figure 9.

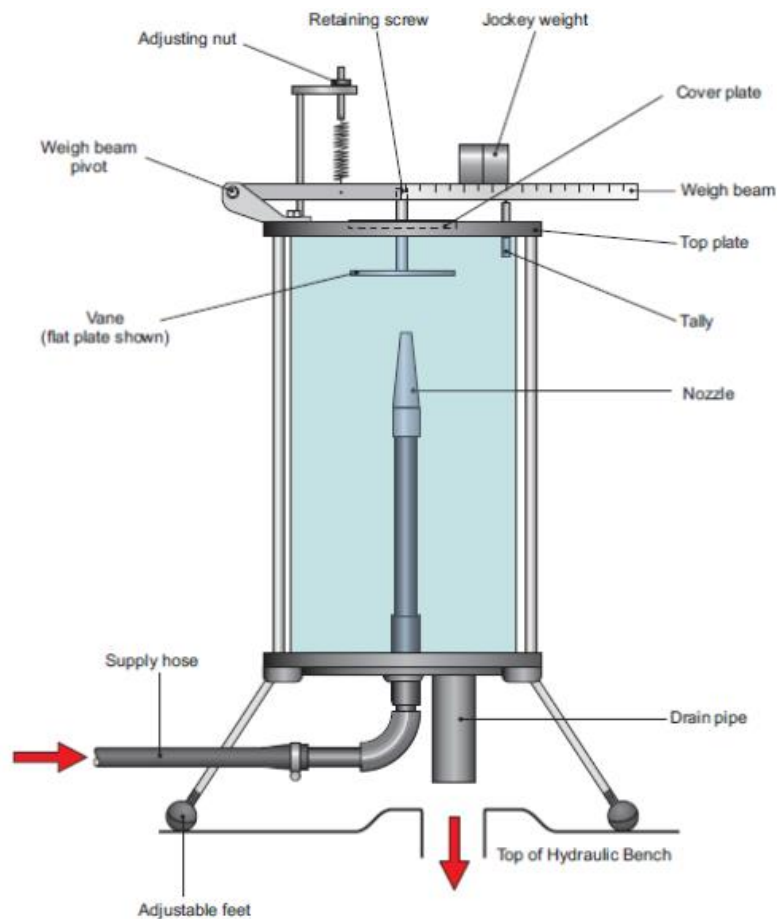


Figure 9. Lay out of the jets on vanes experimental set up

Try

1. Change the vane angle to 45° and repeat the same experiment
2. Change the vane angle to 60° and repeat the same experiment
3. Change the orifice geometry and find the coefficient of impact vanes

12. Exercises on Performance of Centrifugal Pumps

12.1 Centrifugal Pump

Start the pump, operate the valves, note down the readings for pressure head, time (t), calculate the actual discharge, input power, output power, and calculate the efficiency of the centrifugal pump, using the experimental setup, shown in figure 10.

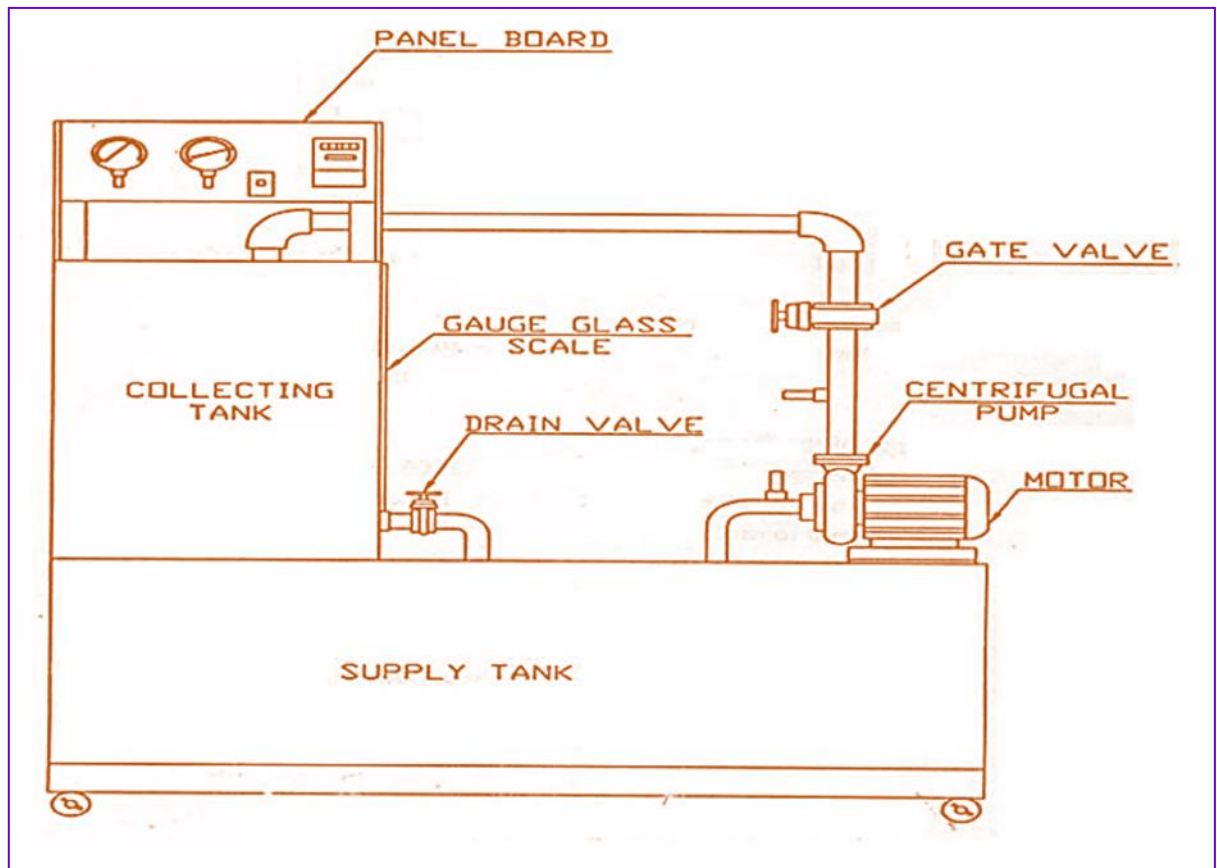


Fig.10. Centrifugal pump experiment setup

Try

1. Note down the time for 15 cm rise of water and calculate the efficiency of the centrifugal pump.
2. Note down the time for 30 cm rise of water and calculate the efficiency of the centrifugal pump.

13. Exercise on Performance of Reciprocating Pump

13.1 Reciprocating Pump

Start the pump, operate the valves, note down the readings for delivery valve, pressure head reading, time (t), calculate the actual discharge, input power, output power, and calculate the efficiency of the centrifugal pump, using the experimental setup, shown in figure 11.

Try

1. Note down the time for 15 cm rise of water and calculate the efficiency of the centrifugal pump
2. Note down the time for 30 cm rise of water and calculate the efficiency of the centrifugal pump

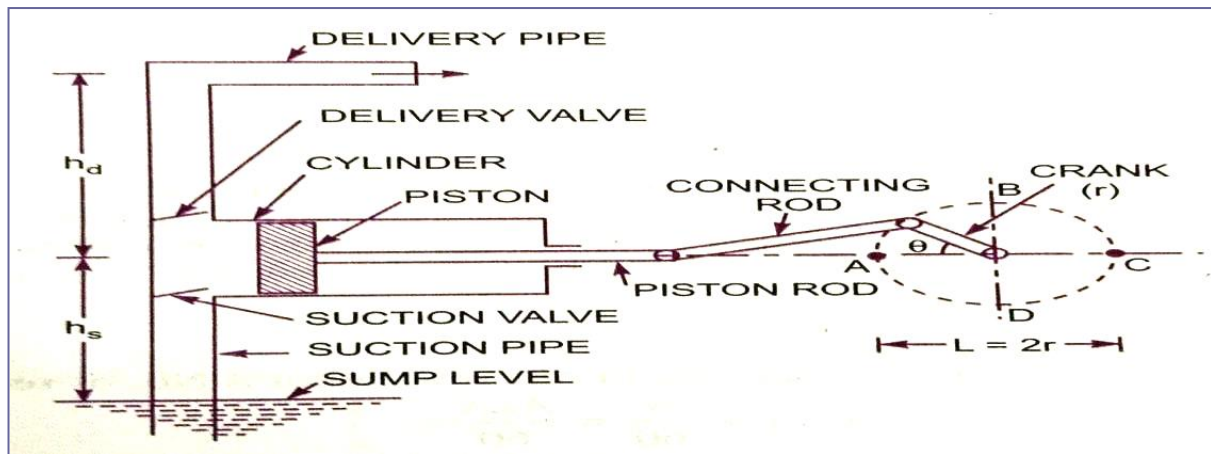


Fig. 11: Reciprocating pump setup

14. Exercise on Pelton Wheel Turbine

14.1 Introduction

Introduction 1. Start the pump, adjust the nozzle opening about half, note down the pressure gauge, vacuum gauge readings, speed of the turbine, manometer readings (h_1 & h_2), and calculate output power, input power, and efficiency of the Pelton wheel turbine, using the experimental setup, shown in figure 12.

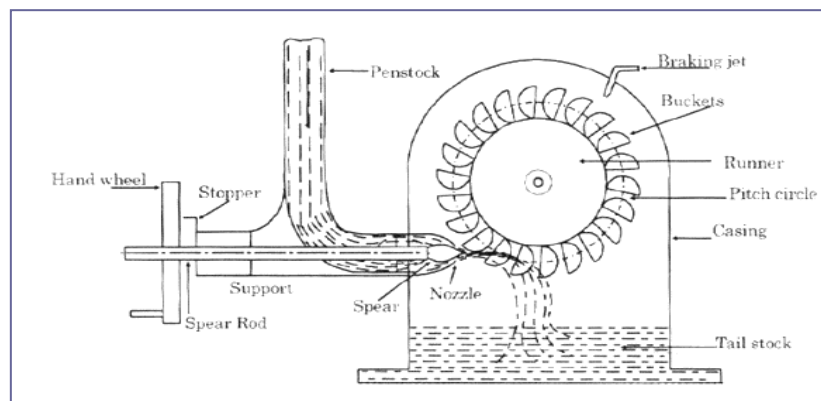


Fig. 12: Schematic diagram of a Pelton turbine

Try

1. Adjust the nozzle opening for full, and calculate the efficiency of the Pelton wheel turbine
2. Note down the time for 30 cm rise of water and calculate the efficiency of the centrifugal pump
3. Performance characteristics of Pelton wheel turbine for change in the bucket design
4. Performance characteristics of Pelton wheel turbine for change in datum head

V. TEXT BOOKS:

1. Gere, Timoshenko, "Mechanics of Materials", McGraw Hill, 3rd edition, 1993.

VI. REFERENCE BOOKS:

1. R. S Kurmi, Gupta, "Strength of Materials", S. Chand, 24th edition, 2005.
2. William Nash, "Strength of Materials", Tata McGraw Hill, 4th edition, 2004.

VII. ELECTRONICS RESOURCES:

1. <https://nptel.ac.in/courses/112107147/>

2. https://vssut.ac.in/lecture_notes/lecture1423904647.pdf
3. <https://web.mit.edu/emech/dontindex-build/>

VIII. MATERIALS ONLINE

1. Course template
2. Lab manual



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

DATA STRUCTURES LABORATORY								
III Semester: AE / ME / CE / ECE / EEE / CSE / CSE (AI & ML) / CSE (DS) / CSE (CS) / IT								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
ACSD11	Core	L	T	P	C	CIA	SEE	Total
		0	0	2	1	40	60	100
Contact Classes: Nil	Tutorial Classes: Nil	Practical Classes: 45			Total Classes: 45			
Prerequisites: There are no prerequisites to take this course.								

I. COURSE OVERVIEW:

The course covers some of the general-purpose data structures and algorithms, and software development. Topics covered include managing complexity, analysis, static data structures, dynamic data structures and hashing mechanisms. The main objective of the course is to teach the students how to select and design data structures and algorithms that are appropriate for problems that they might encounter in real life. This course reaches to student by power point presentations, lecture notes, and lab which involve the problem solving in mathematical and engineering areas.

II. COURSES OBJECTIVES:

The students will try to learn

- To provide students with skills needed to understand and analyze performance trade-offs of different algorithms / implementations and asymptotic analysis of their running time and memory usage.
- To provide knowledge of basic abstract data types (ADT) and associated algorithms: stacks, queues, lists, tree, graphs, hashing and sorting, selection and searching.
- The fundamentals of how to store, retrieve, and process data efficiently.
- To provide practice by specifying and implementing these data structures and algorithms in Python.
- Understand essential for future programming and software engineering courses.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO1 Interpret the complexity of algorithm using the asymptotic notations.
- CO2 Select appropriate searching and sorting technique for a given problem.
- CO3 Construct programs on performing operations on linear and nonlinear data structures for organization of a data.
- CO4 Make use of linear data structures and nonlinear data structures solving real time applications.
- CO5 Describe hashing techniques and collision resolution methods for accessing data with respect to performance.
- CO6 Compare various types of data structures; in terms of implementation, operations and performance.

IV. COURSE CONTENT:

I. LIST OF EXPERIMENTS:

DATA STRUCTURES LABORATORY COURSE CONTENT

S No.	Topic Name
1.	Getting Started Exercises <ul style="list-style-type: none">a. Sum of last digits of two given numbersb. Is N an exact multiple of M?c. Combine Stringsd. Even or Odde. Second last digit of a given numberf. Alternate String Combinerg. Padovan Sequenceh. Leaders in an arrayi. Find the Value of a Number Raised to its Reversej. Mean of Array using Recursion
2.	Searching <ul style="list-style-type: none">a. Linear / Sequential Searchb. Binary Searchc. Uniform Binary Searchd. Interpolation Searche. Fibonacci Search
3.	Sorting <ul style="list-style-type: none">a. Bubble Sortingb. Selection Sortc. Insertion Sort
4.	Divide and Conquer <ul style="list-style-type: none">a. Quick Sortb. Merge Sortc. Heap Sortd. Radix Sorte. Shell Sort
5.	Stack <ul style="list-style-type: none">a. Implementation of Stackb. Balanced Parenthesis Checkingc. Evaluation of Postfix Expressiond. Infix to Postfix Expression Conversione. Reverse a Stack
6.	Queue <ul style="list-style-type: none">a. Linear Queueb. Stack using Queuesc. Queue using Stacksd. Circular Queuee. Deque (Doubly Ended Queue)
7.	Linked List <ul style="list-style-type: none">a. Singly Linked Listb. Linked List Cyclec. Remove Linked List Elements

- d. Reverse Linked List
- e. Palindrome Linked List
- f. Middle of the Linked List
- g. Convert Binary Number in a Linked List to Integer
- 8. Circular Single Linked List and Doubly Linked List**
 - a. Circular Linked List
 - b. Doubly Linked List
 - c. Sorted Merge of Two Sorted Doubly Circular Linked Lists
 - d. Delete all occurrences of a given key in a Doubly Linked List
 - e. Delete a Doubly Linked List Node at a Given Position
- 9. Trees**
 - a. Tree Creation and Basic Tree Terminologies
 - b. Binary Tree Traversal Techniques
 - c. Insertion in a Binary Tree in Level Order
 - d. Finding the Maximum Height or Depth of a Binary Tree
 - e. Deletion in a Binary Tree
- 10. Binary Search Tree (BST)**
 - a. Searching in Binary Search Tree
 - b. Find the node with Minimum Value in a BST
 - c. Check if a Binary Tree is BST or not
 - d. Second Largest Element in BST
 - e. Insertion in Binary Search Tree (BST)
- 11. AVL Tree**
 - a. Insertion in an AVL Tree
 - b. Deletion in an AVL Tree
 - c. Count Greater Nodes in AVL Tree
 - d. Minimum Number of Nodes in an AVL Tree with given Height
- 12. Graph Traversal**
 - a. Breadth First Search
 - b. Depth First Search
 - c. Best First Search (Informed Search)
 - d. Breadth First Traversal of a Graph
 - e. Depth First Search (DFS) for Disconnected Graph
- 13. Minimum Spanning Tree (MST)**
 - a. Kruskal's Algorithm
 - b. Prim's Algorithm
 - c. Total Number of Spanning Trees in a Graph
 - d. Minimum Product Spanning Tree
- 14. Final Notes**

IV. COURSE CONTENT:

EXERCISES FOR DATA STRUCTURES LABORATORY

Note: Students are encouraged to bring their own laptops for laboratory practice sessions.

1. Getting Started Exercises

1.1 Sum of last digits of two given numbers

Rohit wants to add the last digits of two given numbers. For example, If the given numbers are 267 and 154, the output should be 11.

Below is the explanation -

Last digit of the 267 is 7

Last digit of the 154 is 4

Sum of 7 and 4 = 11

Write a program to help Rohit achieve this for any given two numbers.

The prototype of the method should be -

int addLastDigits(int input1, int input2);

where input1 and input2 denote the two numbers whose last digits are to be added.

Note: The sign of the input numbers should be ignored.

if the input numbers are 267 and 154, the sum of last two digits should be 11

if the input numbers are 267 and -154, the sum of last two digits should be 11

if the input numbers are -267 and 154, the sum of last two digits should be 11

if the input numbers are -267 and -154, the sum of last two digits should be 11

Input: 267 154

Output: 11

Input: 267 -154

Output: 11

Input: -267 154

Output: 11

Input: -267 -154

Output: 11

```
import java.util.Scanner;

class AddLastDigitsFunction
{
    int addLastDigits(int n1, int n2)
    {
        # Write code here
    }

    public static void main(String args[])
    {
        AddLastDigitsFunction obj = new AddLastDigitsFunction();
        # Write code here
        System.out.println(obj.addLastDigits(n1,n2));
    }
}
```

1.2 Is N an exact multiple of M?

Write a function that accepts two parameters and finds whether the first parameter is an exact multiple of the second parameter. If the first parameter is an exact multiple of the second parameter, the function should return 2 else it should return 1.

If either of the parameters are zero, the function should return 3.

Assumption: Within the scope of this question, assume that - the first parameter can be positive, negative or zero the second parameter will always be ≥ 0

Input: num1 = 10, num2 = 5

Output: 2

Input: num1 = -10, num2 = 5

Output: 2

Input: num1 = 0, num2 = 5

Output: 3

Input: num1 = 10, num2 = 3

Output: 1

```
public class MultipleChecker
{
    public static int checkMultiple(int num1, int num2)
    {
        # Write code here
    }

    public static void main(String[] args)
    {
        # Write code here
    }
}
```

1.3 Combine Strings

Given 2 strings, a and b, return a new string of the form short+long+short, with the shorter string on the outside and the longer string in the inside. The strings will not be the same length, but they may be empty (length 0).

If input is "hi" and "hello", then output will be "hihellohi"

Input: Enter the first string: "hi"

Enter the second string: "hello"

Output: "hihellohi"

Input: Enter the first string: "iare"

Enter the second string: "college"

Output: "iarecollegeiare"

```
public class StringCombiner
{
    public static void main(String[] args)
    {
        # Write code here
    }

    public static String combineStrings(String a, String b)
    {
        # Write code here
    }
}
```

1.4 Even or Odd

Write a function that accepts 6 input parameters. The first 5 input parameters are of type int. The sixth input parameter is of type string. If the sixth parameter contains the value "even", the function is supposed to return the count of how many of the first five input parameters are even. If the sixth parameter contains the value "odd", the function is supposed to return the count of how many of the first five input parameters are odd.

Example:

If the five input parameters are 12, 17, 19, 14, and 115, and the sixth parameter is "odd", the function must return 3, because there are three odd numbers 17, 19 and 115.

If the five input parameters are 12, 17, 19, 14, and 115, and the sixth parameter is "even", the function must return 2, because there are two even numbers 12 and 14.

Note that zero is considered an even number.

Input: num1 = 12;
num2 = 17;
num3 = 19;
num4 = 14;
num5 = 115;
type = "odd"

Output: 3

Input: num1 = 12;
num2 = 17;
num3 = 19;
num4 = 14;
num5 = 115;
type = "even"

Output: 2

```
public class NumberCounter
{
    public static int countNumbers(int num1, int num2, int num3, int num4, int num5, String type)
    {
        # Write code here
    }

    public static void main(String[] args)
    {
        # Write code here
    }
}
```

1.5 Second last digit of a given number

Write a function that returns the second last digit of the given number. Second last digit is being referred to the digit in the tens place in the given number.

Example: if the given number is 197, the second last digit is 9.

Note 1: The second last digit should be returned as a positive number. i.e. if the given number is -197, the second last digit is 9.

Note 2: If the given number is a single digit number, then the second last digit does not exist. In such cases, the function should return -1. i.e. if the given number is 5, the second last digit should be returned as -1.

Input: 197

Output: 9
Input: 5
Output: -1
Input: -197
Output: 9

```
public class SecondLastDigit
{
    public static int getSecondLastDigit(int number)
    {
        # write code here
    }
    public static void main(String[] args)
    {
        # write code here
    }
}
```

1.6 Alternate String Combiner

Given two strings, a and b, print a new string which is made of the following combination-first character of a, the first character of b, second character of a, second character of b and so on.

Any characters left, will go to the end of the result.

Hello,World

HWeolrlld

Input: "Hello,World"

Output: "HWeolrlld"

Input: "Iare,College"

Output: "ICaorlelege"

```
public class AlternateStringCombiner
{
    public static void main(String[] args)
    {
        # write code here
    }
    public static String combineStrings(String a, String b)
    {
        # write code here
    }
}
```

1.7 Padovan Sequence

The Padovan sequence is a sequence of numbers named after Richard Padovan, who attributed its discovery to Dutch architect Hans van der Laan. The sequence was described by Ian Stewart in his Scientific American column Mathematical Recreations in June 1996. The Padovan sequence is defined by the following recurrence relation:

$$P(n) = P(n-2) + P(n-3)$$

with the initial conditions $P(0) = P(1) = P(2) = 1$.

In this sequence, each term is the sum of the two preceding terms, similar to the Fibonacci sequence. However, the Padovan sequence has different initial conditions and exhibits different growth patterns.

The first few terms of the Padovan sequence are: 1, 1, 1, 2, 2, 3, 4, 5, 7, 9, 12, ...

Input: num = 10

Output: Padovan Sequence up to 10:

1 1 1 2 2 3 4 5 7 9 12

Input: num = 20

Output: Padovan Sequence up to 20:

1 1 1 2 2 3 4 5 7 9 12 16 21 28 37 49 65 86 114 151 200

```
public class PadovanSequence
{
    public static int padovan(int n)
    {
        # write code here
    }
    public static void main(String[] args)
    {
        # write code here
    }
}
```

1.8 Leaders in an array

Given an array arr of n positive integers, your task is to find all the leaders in the array. An element of the array is considered a leader if it is greater than all the elements on its right side or if it is equal to the maximum element on its right side. The rightmost element is always a leader.

Input: n = 6, arr[] = {16, 17, 4, 3, 5, 2}

Output: 17 5 2

Input: n = 5, arr[] = {10, 4, 2, 4, 1}

Output: 10 4 4 1

Input: n = 4, arr[] = {5, 10, 20, 40}

Output: 40

Input: n = 4, arr[] = {30, 10, 10, 5}

Output: 30 10 10 5

```
import java.util.ArrayList;
import java.util.List;
public class ArrayLeaders
{
    public static List<Integer> findArrayLeaders(int[] arr)
    {
        # write code here
    }
    public static void main(String[] args)
    {
        # write code here
    }
}
```

1.9 Find the Value of a Number Raised to its Reverse

Given a number N and its reverse R. The task is to find the number obtained when the number is raised to the power of its own reverse

Input : N = 2, R = 2

Output: 4

Explanation: Number 2 raised to the power of its reverse 2 gives 4 which gives 4 as a result after performing modulo 10^9+7

Input: N = 57, R = 75

Output: 262042770

Explanation: 57^{75} modulo 10^9+7 gives us the result as 262042770

```
public class NumberPower
{
    public static long powerOfReverse(int N, int R)
    {
        # write code here
    }
    public static void main(String[] args)
    {
        # write code here
    }
}
```

1.10 Mean of Array using Recursion

Find the mean of the elements of the array.

Mean = (Sum of elements of the Array) / (Total no of elements in Array)

Input: 1 2 3 4 5

Output: 3.0

Input: 1 2 3

Output: 2.0

To find the mean using recursion assume that the problem is already solved for N-1 i.e. you have to find for n

Sum of first N-1 elements = (Mean of N-1 elements) * (N-1)

Mean of N elements = (Sum of first N-1 elements + N-th elements) / (N)

```
public class ArrayMean
{
    public static double findArrayMean(int[] arr)
    {
        # write code here
    }
    public static void main(String[] args)
    {
        # write code here
    }
}
```

Try:

1. **Kth Smallest Element:** Given an array arr[] and an integer k where k is smaller than the size of the array, the task is to find the kth smallest element in the given array. It is given that all array elements are distinct.

Note: l and r denotes the starting and ending index of the array.

Input: n = 6, arr[] = {7, 10, 4, 3, 20, 15}, k = 3, l = 0, r = 5

Output: 7

Explanation: 3rd smallest element in the given array is 7.

Input: n = 5, arr[] = {7, 10, 4, 20, 15}, k = 4, l=0 r=4

Output: 15

Explanation: 4th smallest element in the given array is 15.

Your task is to complete the function **kthSmallest()** which takes the array `arr[]`, integers `l` and `r` denoting the starting and ending index of the array and an integer `k` as input and returns the `k`th smallest element.

2. **Count pairs with given sum:** Given an array of `N` integers, and an integer `K`, find the number of pairs of elements in the array whose sum is equal to `K`. Your task is to complete the function **getPairsCount()** which takes `arr[]`, `n` and `k` as input parameters and returns the number of pairs that have sum `K`.

Input: `N = 4, K = 6, arr[] = {1, 5, 7, 1}`

Output: 2

Explanation: `arr[0] + arr[1] = 1 + 5 = 6` and `arr[1] + arr[3] = 5 + 1 = 6`.

Input: `N = 4, K = 2, arr[] = {1, 1, 1, 1}`

Output: 6

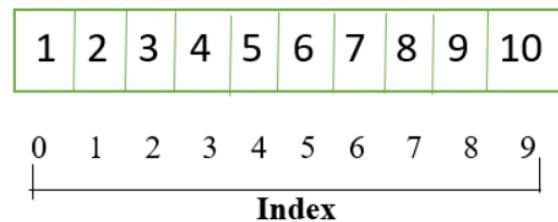
Explanation: Each 1 will produce sum 2 with any 1.

2. Searching

2.1 Linear / Sequential Search

Linear search is defined as the searching algorithm where the list or data set is traversed from one end to find the desired value. Given an array `arr[]` of `n` elements, write a recursive function to search a given element `x` in `arr[]`.

Find '6'



1	2	3	4	5	6	7	8	9	10
0	1	2	3	4	5	6	7	8	9
Index									

Note : We find '6' at index '5' through linear search

Linear search procedure:

1. Start from the leftmost element of `arr[]` and one by one compare `x` with each element of `arr[]`
2. If `x` matches with an element, return the index.
3. If `x` doesn't match with any of the elements, return -1.

Input: `arr[] = {10, 20, 80, 30, 60, 50, 110, 100, 130, 170}`

`x = 110;`

Output: 6

Element `x` is present at index 6

Input: `arr[] = {10, 20, 80, 30, 60, 50, 110, 100, 130, 170}`

`x = 175;`

Output: -1

Element `x` is not present in `arr[]`.

```
public class RecursiveLinearSearch
{
    public static int recursiveLinearSearch(int[] arr, int key, int index)
    {
        # write code here
    }
}
```

```

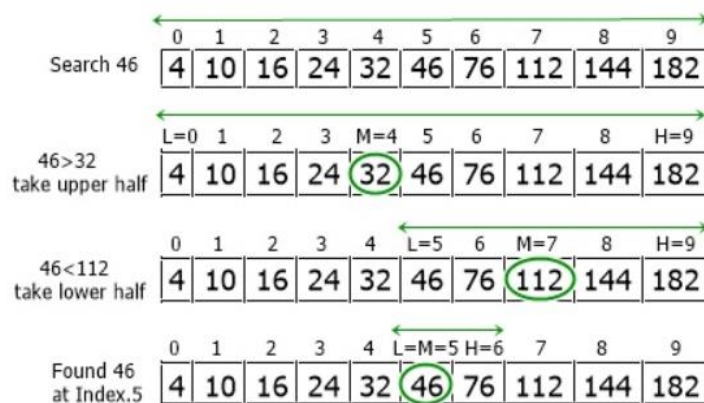
    }

    public static void main(String[] args)
    {
        # write code here
    }
}

```

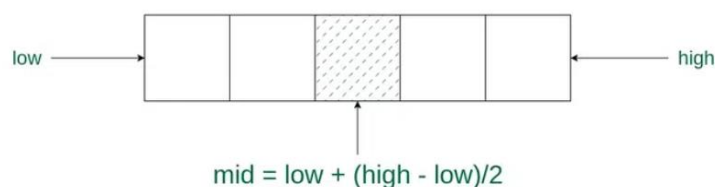
2.2 Binary Search

Binary Search is defined as a searching algorithm used in a sorted array by repeatedly dividing the search interval in half. The idea of binary search is to use the information that the array is sorted and reduce the time complexity to $O(\log N)$.



Conditions for Binary Search algorithm:

1. The data structure must be sorted.
2. Access to any element of the data structure takes constant time.



Binary Search Procedure:

1. Divide the search space into two halves by finding the middle index "mid".
2. Compare the middle element of the search space with the key.
3. If the key is found at middle element, the process is terminated.
4. If the key is not found at middle element, choose which half will be used as the next search space.
 - a. If the key is smaller than the middle element, then the left side is used for next search.
 - b. If the key is larger than the middle element, then the right side is used for next search.
5. This process is continued until the key is found or the total search space is exhausted.

Input: arr = [2, 5, 8, 12, 16, 23, 38, 56, 72, 91]

Output: target = 23

Element 23 is present at index 5

```

public class RecursiveBinarySearch
{

```

```

    public static int recursiveBinarySearch(int[] arr, int key, int left, int
right)
    {
        # write code here
    }

    public static void main(String[] args)
    {
        # write code here
    }
}

```

2.3 Uniform Binary Search

Uniform Binary Search is an optimization of Binary Search algorithm when many searches are made on same array or many arrays of same size. In normal binary search, we do arithmetic operations to find the mid points. Here we precompute mid points and fills them in lookup table. The array look-up generally works faster than arithmetic done (addition and shift) to find the mid-point.

Input: array = {1, 3, 5, 6, 7, 8, 9}, v=3

Output: Position of 3 in array = 2

Input: array = {1, 3, 5, 6, 7, 8, 9}, v=7

Output: Position of 7 in array = 5

The algorithm is very similar to Binary Search algorithm, the only difference is a lookup table is created for an array and the lookup table is used to modify the index of the pointer in the array which makes the search faster. Instead of maintaining lower and upper bound the algorithm maintains an index and the index is modified using the lookup table.

```

public class RecursiveUniformBinarySearch
{
    public static int recursiveUniformBinarySearch(int[] arr, int key, int[]
lookupTable, int left, int right)
    {
        # write code here
    }
    public static void main(String[] args)
    {
        # write code here
    }
}

```

2.4 Interpolation Search

Interpolation search works better than Binary Search for a Sorted and Uniformly Distributed array. Binary search goes to the middle element to check irrespective of search-key. On the other hand, Interpolation search may go to different locations according to search-key. If the value of the search-key is close to the last element, Interpolation Search is likely to start search toward the end side. Interpolation search is more efficient than binary search when the elements in the list are uniformly distributed, while binary search is more efficient when the elements in the list are not uniformly distributed.

Interpolation search can take longer to implement than binary search, as it requires the use of additional calculations to estimate the position of the target element.

Input: arr = [1, 2, 3, 4, 5, 6, 7, 8, 9]

Output: target = 5

```
public class InterpolationSearch
{
    public static int interpolationSearch(int[] arr, int key)
    {
        # write code here
    }

    public static void main(String[] args)
    {
        # write code here
    }
}
```

2.5 Fibonacci Search

Given a sorted array arr[] of size n and an element x to be searched in it. Return index of x if it is present in array else return -1.

Input: arr[] = {2, 3, 4, 10, 40}, x = 10

Output: 3

Element x is present at index 3.

Input: arr[] = {2, 3, 4, 10, 40}, x = 11

Output: -1

Element x is not present.

Fibonacci Search is a comparison-based technique that uses Fibonacci numbers to search an element in a sorted array.

Fibonacci Numbers are recursively defined as $F(n) = F(n-1) + F(n-2)$, $F(0) = 0$, $F(1) = 1$. First few Fibonacci Numbers are 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

Fibonacci Search Procedure:

Let the searched element be x. The idea is to first find the smallest Fibonacci number that is greater than or equal to the length of the given array. Let the found Fibonacci number be fib (m'th Fibonacci number). We use (m-2)'th Fibonacci number as the index (If it is a valid index). Let (m-2)'th Fibonacci Number be i, we compare arr[i] with x, if x is same, we return i. Else if x is greater, we recur for subarray after i, else we recur for subarray before i.

Let arr[0..n-1] be the input array and the element to be searched be x.

1. Find the smallest Fibonacci number greater than or equal to n. Let this number be fibM [m'th Fibonacci number]. Let the two Fibonacci numbers preceding it be fibMm1 [(m-1)'th Fibonacci Number] and fibMm2 [(m-2)'th Fibonacci Number].
2. While the array has elements to be inspected:
 - i. Compare x with the last element of the range covered by fibMm2
 - ii. If x matches, return index
 - iii. Else If x is less than the element, move the three Fibonacci variables two Fibonacci down, indicating elimination of approximately rear two-third of the remaining array.
 - iv. Else x is greater than the element, move the three Fibonacci variables one Fibonacci down. Reset offset to index. Together these indicate the elimination of approximately front one-third of the remaining array.

3. Since there might be a single element remaining for comparison, check if fibMm1 is 1. If Yes, compare x with that remaining element. If match, return index.

```
public class FibonacciSearch
{
    public static int fibonacciSearch(int[] arr, int key)
    {
        # write code here
    }

    public static void main(String[] args)
    {
        # write code here
    }
}
```

3. Sorting

3.1 Bubble Sort

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in the wrong order. This algorithm is not suitable for large data sets as its average and worst-case time complexity is quite high.

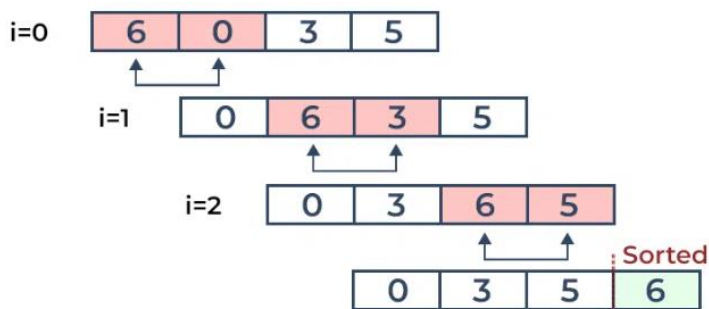
Bubble Sort Procedure:

1. Traverse from left and compare adjacent elements and the higher one is placed at right side.
2. In this way, the largest element is moved to the rightmost end at first.
3. This process is then continued to find the second largest and place it and so on until the data is sorted.

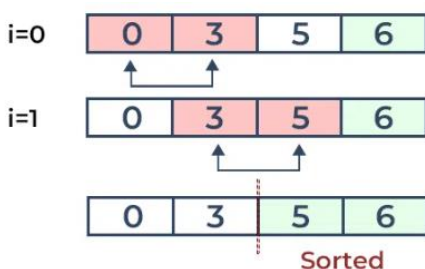
Input: arr = [6, 3, 0, 5]

Output:

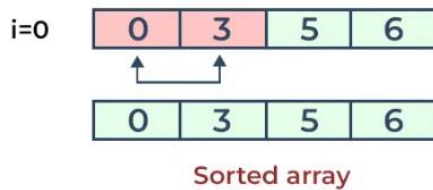
First Pass:



Second Pass:



Third Pass:



```
import java.util.Scanner;

class BubbleSortExample
{
    public static void main(String[] args)
    {
        # write code here
    }

    public static void bubbleSort(int[] arr)
    {
        # write code here
    }
}
```

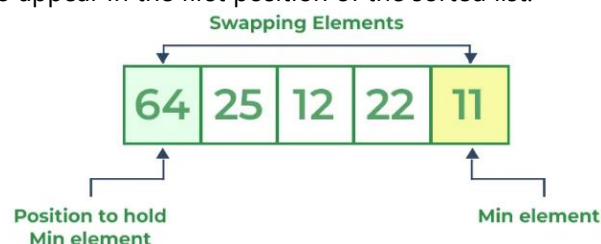
3.2 Selection Sort

Selection sort is a simple and efficient sorting algorithm that works by repeatedly selecting the smallest (or largest) element from the unsorted portion of the list and moving it to the sorted portion of the list. The algorithm repeatedly selects the smallest (or largest) element from the unsorted portion of the list and swaps it with the first element of the unsorted part. This process is repeated for the remaining unsorted portion until the entire list is sorted.

Input: arr = [64, 25, 12, 22, 11]

Output: arr = [11, 12, 22, 25, 64]

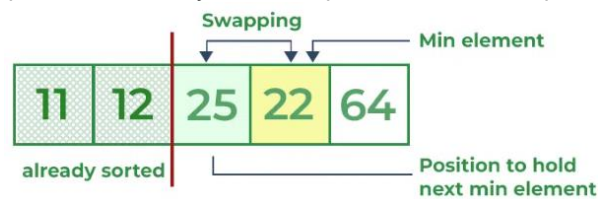
First Pass: For the first position in the sorted array, the whole array is traversed from index 0 to 4 sequentially. The first position where 64 is stored presently, after traversing whole array it is clear that 11 is the lowest value. Thus, replace 64 with 11. After one iteration 11, which happens to be the least value in the array, tends to appear in the first position of the sorted list.



Second Pass: For the second position, where 25 is present, again traverse the rest of the array in a sequential manner. After traversing, we found that 12 is the second lowest value in the array and it should appear at the second place in the array, thus swap these values.



Third Pass: Now, for third place, where 25 is present again traverse the rest of the array and find the third least value present in the array. While traversing, 22 came out to be the third least value and it should appear at the third place in the array, thus swap 22 with element present at third position.



Fourth Pass: Similarly, for fourth position traverse the rest of the array and find the fourth least element in the array. As 25 is the 4th lowest value hence, it will place at the fourth position.



Fifth Pass: At last the largest value present in the array automatically get placed at the last position in the array. The resulted array is the sorted array.



```
import java.util.Scanner;

class SelectionSortExample
{
    public static void main(String[] args)
    {
        # write code here
    }

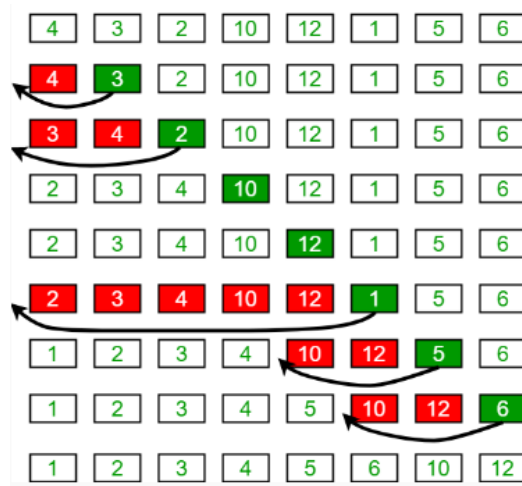
    public static void selectionSort(int[] arr)
    {
        # write code here
    }
}
```

3.3 Insertion Sort

Insertion sort is a simple sorting algorithm that works similar to the way you sort playing cards in your hands. The array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed at the correct position in the sorted part.

Insertion Sort Procedure:

1. To sort an array of size N in ascending order iterate over the array and compare the current element (key) to its predecessor, if the key element is smaller than its predecessor, compare it to the elements before.
2. Move the greater elements one position up to make space for the swapped element.



Input: arr = [4, 3, 2, 10, 12, 1, 5, 6]

Output: arr = [1, 2, 3, 4, 5, 6, 10, 12]

```
import java.util.Scanner;

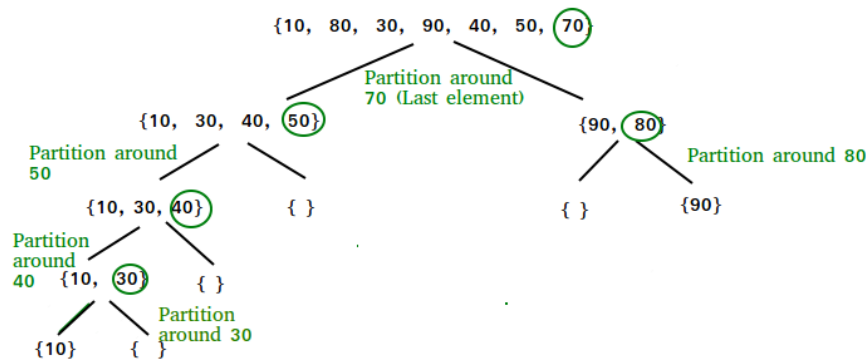
class InsertionSortExample
{
    public static void main(String[] args)
    {
        # write code here
    }

    public static void insertionSort(int[] arr)
    {
        # write code here
    }
}
```

4. Divide and Conquer

4.1 Quick Sort

QuickSort is a sorting algorithm based on the Divide and Conquer algorithm that picks an element as a pivot and partitions the given array around the picked pivot by placing the pivot in its correct position in the sorted array. The key process in quickSort is a partition(). The target of partitions is to place the pivot (any element can be chosen to be a pivot) at its correct position in the sorted array and put all smaller elements to the left of the pivot, and all greater elements to the right of the pivot. Partition is done recursively on each side of the pivot after the pivot is placed in its correct position and this finally sorts the array.



The quick sort method can be summarized in three steps:

1. **Pick:** Select a pivot element.
2. **Divide:** Split the problem set, move smaller parts to the left of the pivot and larger items to the right.
3. **Repeat and combine:** Repeat the steps and combine the arrays that have previously been sorted.

Algorithm for Quick Sort Function:

```
//start --> Starting index, end --> Ending index
Quicksort(array, start, end)
{
    if (start < end)
    {
        pIndex = Partition(A, start, end)
        Quicksort(A, start, pIndex-1)
        Quicksort(A, pIndex+1, end)
    }
}
```

Algorithm for Partition Function:

```
partition (array, start, end)
{
    // Setting rightmost Index as pivot
    pivot = arr[end];

    i = (start - 1) // Index of smaller element and indicates the
    // right position of pivot found so far
    for (j = start; j <= end- 1; j++)
    {
        // If current element is smaller than the pivot
        if (arr[j] < pivot)
        {
            i++; // increment index of smaller element
            swap arr[i] and arr[j]
        }
    }
    swap arr[i + 1] and arr[end])
    return (i + 1)
}
```

Input: arr = [10, 80, 30, 90, 40, 50, 70]
Output: arr = [10, 30, 40, 50, 70, 80, 90]

```
import java.util.Scanner;

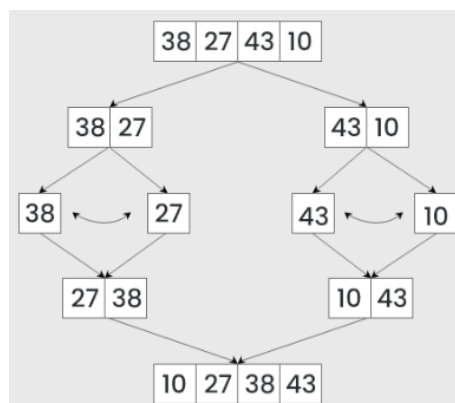
class QuickSortExample
{
    public static void main(String[] args)
    {
        # write code here
    }

    public static void quickSort(int[] arr, int low, int high)
    {
        # write code here
    }

    public static int partition(int[] arr, int low, int high)
    {
        # write code here
    }
}
```

4.2 Merge Sort

Merge sort is defined as a sorting algorithm that works by dividing an array into smaller subarrays, sorting each subarray, and then merging the sorted subarrays back together to form the final sorted array. In simple terms, we can say that the process of merge sort is to divide the array into two halves, sort each half, and then merge the sorted halves back together. This process is repeated until the entire array is sorted.



Input: arr = [12, 11, 13, 5, 6, 7]
Output: arr = [5, 6, 7, 11, 12, 13]

```
import java.util.Scanner;

class MergeSortExample
{
    public static void main(String[] args)
    {
        # write code here
    }
}
```

```

    public static void mergeSort(int[] arr, int low, int high)
    {
        # write code here
    }

    public static void merge(int[] arr, int low, int mid, int high)
    {
        # write code here
    }
}

```

4.3 Heap Sort

Heap sort is a comparison-based sorting technique based on Binary Heap data structure. It is similar to the selection sort where we first find the minimum element and place the minimum element at the beginning. Repeat the same process for the remaining elements.

Heap Sort Procedure:

First convert the array into heap data structure using heapify, then one by one delete the root node of the Max-heap and replace it with the last node in the heap and then heapify the root of the heap. Repeat this process until size of heap is greater than 1.

- Build a heap from the given input array.
- Repeat the following steps until the heap contains only one element:
 - Swap the root element of the heap (which is the largest element) with the last element of the heap.
 - Remove the last element of the heap (which is now in the correct position).
 - Heapify the remaining elements of the heap.
- The sorted array is obtained by reversing the order of the elements in the input array.

Input: arr = [12, 11, 13, 5, 6, 7]

Output: Sorted array is 5 6 7 11 12 13

```

import java.util.Scanner;

class HeapSortExample
{
    public static void main(String[] args)
    {
        # write code here
    }

    public static void heapSort(int[] arr)
    {
        # write code here
    }

    public static void heapify(int[] arr, int n, int i)
    {
        # write code here
    }
}

```

4.4 Radix Sort

Radix Sort is a linear sorting algorithm that sorts elements by processing them digit by digit. It is an efficient sorting algorithm for integers or strings with fixed-size keys. Rather than comparing elements directly, Radix Sort distributes the elements into buckets based on each digit's value. By repeatedly sorting the elements by their significant digits, from the least significant to the most significant, Radix Sort achieves the final sorted order.

Radix Sort Procedure:

The key idea behind Radix Sort is to exploit the concept of place value.

1. It assumes that sorting numbers digit by digit will eventually result in a fully sorted list.
2. Radix Sort can be performed using different variations, such as Least Significant Digit (LSD) Radix Sort or Most Significant Digit (MSD) Radix Sort.

To perform radix sort on the array [170, 45, 75, 90, 802, 24, 2, 66], we follow these steps:



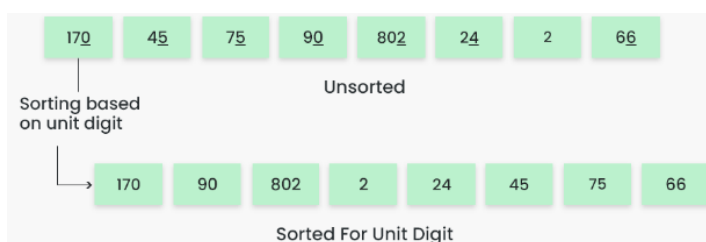
Step 1: Find the largest element in the array, which is 802. It has three digits, so we will iterate three times, once for each significant place.

Step 2: Sort the elements based on the unit place digits ($X=0$). We use a stable sorting technique, such as counting sort, to sort the digits at each significant place.

Sorting based on the unit place:

Perform counting sort on the array based on the unit place digits.

The sorted array based on the unit place is [170, 90, 802, 2, 24, 45, 75, 66]

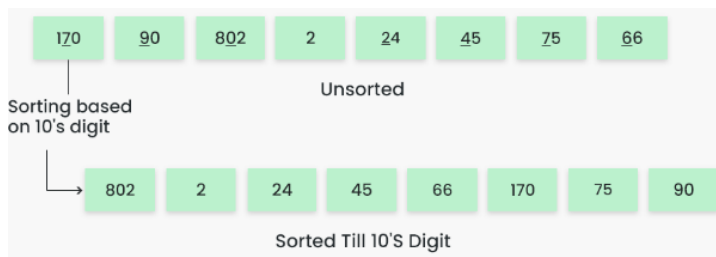


Step 3: Sort the elements based on the tens place digits.

Sorting based on the tens place:

Perform counting sort on the array based on the tens place digits.

The sorted array based on the tens place is [802, 2, 24, 45, 66, 170, 75, 90]

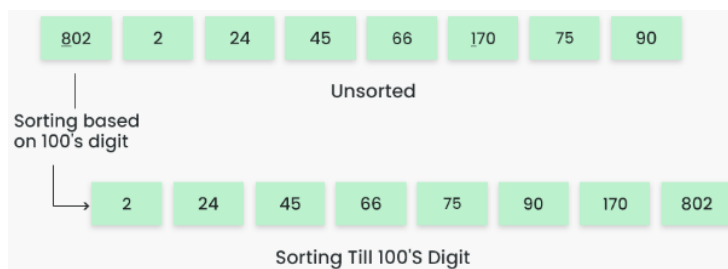


Step 4: Sort the elements based on the hundreds place digits.

Sorting based on the hundreds place:

Perform counting sort on the array based on the hundreds place digits.

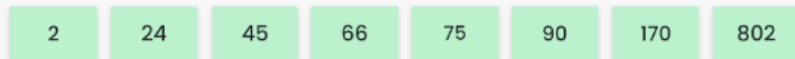
The sorted array based on the hundreds place is [2, 24, 45, 66, 75, 90, 170, 802]



Step 5: The array is now sorted in ascending order.

The final sorted array using radix sort is [2, 24, 45, 66, 75, 90, 170, 802]

Array after performing Radix Sort for all digits



```
import java.util.Arrays;

class RadixSortExample
{
    public static void main(String[] args)
    {
        # write code here
    }

    public static void radixSort(int[] arr)
    {
        # write code here
    }

    public static int getMax(int[] arr)
    {
        # write code here
    }

    public static void countSort(int[] arr, int exp)
    {
        # write code here
    }
}
```

```
}
```

4.5 Shell Sort

Shell sort is mainly a variation of Insertion Sort. In insertion sort, we move elements only one position ahead. When an element has to be moved far ahead, many movements are involved. The idea of ShellSort is to allow the exchange of far items. In Shell sort, we make the array h-sorted for a large value of h. We keep reducing the value of h until it becomes 1. An array is said to be h-sorted if all sublists of every h'th element are sorted.

Shell Sort Procedure:

1. Initialize the value of gap size h
2. Divide the list into smaller sub-part. Each must have equal intervals to h
3. Sort these sub-lists using insertion sort
4. Repeat this step 1 until the list is sorted.
5. Print a sorted list.

```
Procedure Shell_Sort(Array, N)
  While Gap < Length(Array) /3 :
    Gap = ( Interval * 3 ) + 1
  End While Loop
  While Gap > 0 :
    For (Outer = Gap; Outer < Length(Array); Outer++):
      Insertion_Value = Array[Outer]
      Inner = Outer;
      While Inner > Gap-1 And Array[Inner - Gap] >= Insertion_Value:
        Array[Inner] = Array[Inner - Gap]
        Inner = Inner - Gap
      End While Loop
      Array[Inner] = Insertion_Value
    End For Loop
    Gap = (Gap -1) /3;
  End While Loop
End Shell_Sort
```

```
import java.util.Scanner;

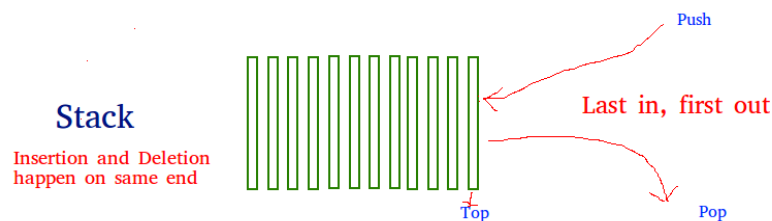
class ShellSortExample
{
    public static void main(String[] args)
    {
        # write code here
    }

    public static void shellSort(int[] arr)
    {
        # write code here
    }
}
```


5. Stack

5.1 Implementation of Stack

A stack is a linear data structure that stores items in a Last-In/First-Out (LIFO) or First-In/Last-Out (FILO) manner. In stack, a new element is added at one end and an element is removed from that end only. The insert and delete operations are often called push and pop.



The functions associated with stack are:

- **empty()** – Returns whether the stack is empty
- **size()** – Returns the size of the stack
- **top() / peek()** – Returns a reference to the topmost element of the stack
- **push(a)** – Inserts the element 'a' at the top of the stack
- **pop()** – Deletes the topmost element of the stack

```
class Stack
{
    private int maxSize;
    private int top;
    private int[] stackArray;

    public Stack(int size)
    {
        # write code here
    }

    public void push(int value)
    {
        # write code here
    }

    public int pop()
    {
        # write code here
    }

    public int peek()
    {
        # write code here
    }

    public boolean isEmpty()
    {
        # write code here
    }

    public boolean isFull()
    {

```

```

        # write code here
    }
}

class StackExample
{
    public static void main(String[] args)
    {
        Stack stack = new Stack(5);
        stack.push(10);
        stack.push(20);
        stack.push(30);
        stack.pop();
        stack.peek();
        stack.push(40);
        stack.push(50);
        stack.push(60);
    }
}

```

5.2 Balanced Parenthesis Checking

Given an expression string, write a java program to find whether a given string has balanced parentheses or not.

Input: "{(a+b)*(c-d)}"

Output: true

Input: "{(a+b)*[c-d]}"

Output: false

One approach to check balanced parentheses is to use stack. Each time, when an open parentheses is encountered push it in the stack, and when closed parenthesis is encountered, match it with the top of stack and pop it. If stack is empty at the end, return true otherwise, false

```

import java.util.Stack;

class BalancedParenthesisChecker
{
    public static boolean isBalanced(String expression)
    {
        # write code here
    }

    public static void main(String[] args)
    {
        String expression1 = "{(a+b)*(c-d)}";
        String expression2 = "{(a+b)*[c-d]}";

        # write code here
    }
}

```

5.3 Evaluation of Postfix Expression

Given a postfix expression, the task is to evaluate the postfix expression. Postfix expression: The expression of the form "a b operator" (ab+) i.e., when a pair of operands is followed by an operator.

Input: str = "2 3 1 * + 9 -"

Output: -4

Explanation: If the expression is converted into an infix expression, it will be $2 + (3 * 1) - 9 = 5 - 9 = -4$.

Input: str = "100 200 + 2 / 5 * 7 +"

Output: 757

Procedure for evaluation postfix expression using stack:

- Create a stack to store operands (or values).
- Scan the given expression from left to right and do the following for every scanned element.
 - If the element is a number, push it into the stack.
 - If the element is an operator, pop operands for the operator from the stack. Evaluate the operator and push the result back to the stack.
- When the expression is ended, the number in the stack is the final answer.

```
import java.util.Stack;

class PostfixEvaluator
{
    public static int evaluatePostfix(String expression)
    {
        # write code here
    }

    public static int performOperation(char operator, int operand1, int operand2)
    {
        # write code here
    }

    public static void main(String[] args)
    {
        # write code here
    }
}
```

5.4 Infix to Postfix Expression Conversion

For a given Infix expression, convert it into Postfix form.

Infix expression: The expression of the form "a operator b" (a + b) i.e., when an operator is in-between every pair of operands.

Postfix expression: The expression of the form "a b operator" (ab+) i.e., When every pair of operands is followed by an operator.

Infix to postfix expression conversion procedure:

1. Scan the infix expression from left to right.
2. If the scanned character is an operand, put it in the postfix expression.
3. Otherwise, do the following
 - If the precedence and associativity of the scanned operator are greater than the precedence and associativity of the operator in the stack [or the stack is empty or the stack contains a '('], then push it in the stack. ['^' operator is right associative and other operators like '+', '-', '*', and '/' are left-associative].
 - Check especially for a condition when the operator at the top of the stack and the scanned operator both are '^'. In this condition, the precedence of the scanned operator is higher due to its right associativity. So it will be pushed into the operator stack.
 - In all the other cases when the top of the operator stack is the same as the scanned operator, then pop the operator from the stack because of left associativity due to which the scanned operator has less precedence.

- Else, Pop all the operators from the stack which are greater than or equal to in precedence than that of the scanned operator.
 - After doing that Push the scanned operator to the stack. (If you encounter parenthesis while popping then stop there and push the scanned operator in the stack.)
4. If the scanned character is a '(', push it to the stack.
 5. If the scanned character is a ')', pop the stack and output it until a '(' is encountered, and discard both the parenthesis.
 6. Repeat steps 2-5 until the infix expression is scanned.
 7. Once the scanning is over, Pop the stack and add the operators in the postfix expression until it is not empty.
 8. Finally, print the postfix expression.

Input: A + B * C + D

Output: A B C * + D +

Input: ((A + B) - C * (D / E)) + F

Output: A B + C D E / * - F +

```
import java.util.Stack;
```

```
class Conversion
```

```
{
```

```
    # Write Code Here
```

```
}
```

5.5 Reverse a Stack

The stack is a linear data structure which works on the LIFO concept. LIFO stands for last in first out. In the stack, the insertion and deletion are possible at one end the end is called the top of the stack. Define two recursive functions BottomInsertion() and Reverse() to reverse a stack using Python. Define some basic function of the stack like push(), pop(), show(), empty(), for basic operation like respectively append an item in stack, remove an item in stack, display the stack, check the given stack is empty or not.

BottomInsertion(): this method append element at the bottom of the stack and BottomInsertion accept two values as an argument first is stack and the second is elements, this is a recursive method.

Reverse(): the method is reverse elements of the stack, this method accept stack as an argument Reverse() is also a Recursive() function. Reverse() is invoked BottomInsertion() method for completing the reverse operation on the stack.

Input: Elements = [1, 2, 3, 4, 5]

Output: Original Stack

5

4

3

2

1

Stack after Reversing

1

2

3

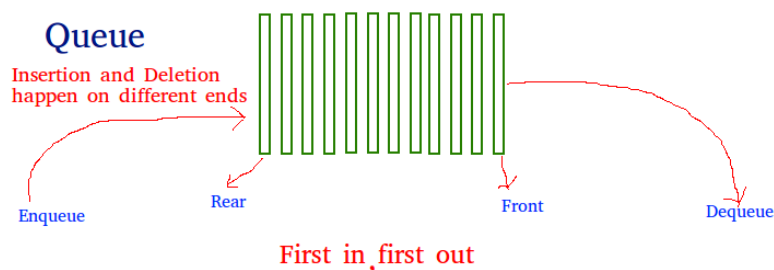
4

```
import java.util.Stack;
class StackClass {
    # Write Code Here
}
```

6. Queue

6.1 Linear Queue

Linear queue is a linear data structure that stores items in First in First out (FIFO) manner. With a queue the least recently added item is removed first. A good example of queue is any queue of consumers for a resource where the consumer that came first is served first.



```
import java.util.Scanner;

public class LinearQueue
{
    # Write Code Here
}

public static boolean isEmpty() {
    return front == rear;
}

public static boolean isFull() {
    return rear == MAX;
}

public static void enqueue(int item)
{
    # Write Code Here
}

public static void dequeue()
{
    # Write Code Here
}

public static void display()
{
    # Write Code Here
}

public static void main(String[] args)
{
    # Write Code Here
}
```

6.2 Stack using Queues

Implement a last-in-first-out (LIFO) stack using only two queues. The implemented stack should support all the functions of a normal stack (push, top, pop, and empty).

- void push(int x) Pushes element x to the top of the stack.
- int pop() Removes the element on the top of the stack and returns it.
- int top() Returns the element on the top of the stack.
- boolean empty() Returns true if the stack is empty, false otherwise.

Input:

["MyStack", "push", "push", "top", "pop", "empty"]

[], [1], [2], [], [], []

Output:

[null, null, null, 2, 2, false]

```
import java.util.LinkedList;
import java.util.Queue;

class MyStack
{
    # Write Code Here
}

public void push(int x)
{
    # Write Code Here
}

public int pop()
{
    return queue.remove();
}

public int top() {
    return queue.peek();
}

public boolean empty() {
    return queue.isEmpty();
}

public static void main(String[] args)
{
    # Write Code Here
}
```

6.3 Queue using Stacks

Implement a first in first out (FIFO) queue using only two stacks. The implemented queue should support all the functions of a normal queue (push, peek, pop, and empty).

- void push(int x) Pushes element x to the back of the queue.
- int pop() Removes the element from the front of the queue and returns it.
- int peek() Returns the element at the front of the queue.
- boolean empty() Returns true if the queue is empty, false otherwise.

Input:

["MyQueue", "push", "push", "peek", "pop", "empty"]

[], [1], [2], [], [], []

Output:

[null, null, null, 1, 1, false]

```
import java.util.Stack;

class MyQueue {

    private Stack<Integer> stack1;
    private Stack<Integer> stack2;
    public MyQueue() {
        stack1 = new Stack<>();
        stack2 = new Stack<>();
    }
    public void push(int x) {
        stack1.push(x);
    }
    public int pop()
    {
        # Write Code Here
    }
    public int peek()
    {
        # Write Code Here
    }
    public boolean empty() {
        return stack1.isEmpty() && stack2.isEmpty();
    }
    public static void main(String[] args)
    {
        # Write Code Here
    }
}
```

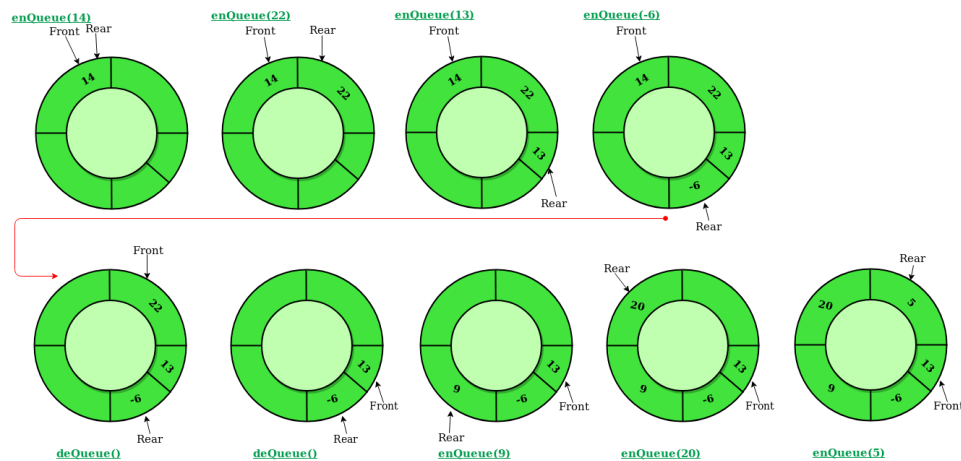
6.4 Circular Queue

A Circular Queue is an extended version of a normal queue where the last element of the queue is connected to the first element of the queue forming a circle. The operations are performed based on FIFO (First In First Out) principle. It is also called 'Ring Buffer'.

Operations on Circular Queue:

- **Front:** Get the front item from the queue.
- **Rear:** Get the last item from the queue.
- **enQueue(value)** This function is used to insert an element into the circular queue. In a circular queue, the new element is always inserted at the rear position.
 - Check whether the queue is full – [i.e., the rear end is in just before the front end in a circular manner].
 - If it is full then display Queue is full.
 - If the queue is not full then, insert an element at the end of the queue.
- **deQueue()** This function is used to delete an element from the circular queue. In a circular queue, the element is always deleted from the front position.
 - Check whether the queue is Empty.
 - If it is empty then display Queue is empty.

- If the queue is not empty, then get the last element and remove it from the queue.



Implement Circular Queue using Array:

5. Initialize an array queue of size **n**, where **n** is the maximum number of elements that the queue can hold.
6. Initialize two variables **front** and **rear** to -1.
7. **Enqueue:** To enqueue an element **x** into the queue, do the following:
 - Increment **rear** by 1.
 - If **rear** is equal to **n**, set **rear** to 0.
 - If **front** is -1, set **front** to 0.
 - Set **queue[rear]** to **x**.
8. **Dequeue:** To dequeue an element from the queue, do the following:
 - Check if the queue is empty by checking if **front** is -1.
 - If it is, return an error message indicating that the queue is empty.
 - Set **x** to **queue[front]**.
 - If **front** is equal to **rear**, set **front** and **rear** to -1.
 - Otherwise, increment **front** by 1 and if **front** is equal to **n**, set **front** to 0.
 - Return **x**.

```
class CircularQueue {
    private int size;
    private int front, rear;
    private int[] queue;

    public CircularQueue(int size) {
        this.size = size;
        this.queue = new int[size];
        this.front = this.rear = -1;
    }

    public void enqueue(int data)
    {
```



```

        # Write Code Here
    }
    public int dequeue()
    {
        # Write Code Here
    }
    public void display()
    {
        # Write Code Here
    }
    public static void main(String[] args)
    {
        # Write Code Here
    }
}

```

6.5 Deque (Doubly Ended Queue)

In a Deque (Doubly Ended Queue), one can perform insert (append) and delete (pop) operations from both the ends of the container. There are two types of Deque:

1. **Input Restricted Deque:** Input is limited at one end while deletion is permitted at both ends.
2. **Output Restricted Deque:** Output is limited at one end but insertion is permitted at both ends.

Operations on Deque:

1. **append():** This function is used to insert the value in its argument to the right end of the deque.
2. **appendleft():** This function is used to insert the value in its argument to the left end of the deque.
3. **pop():** This function is used to delete an argument from the right end of the deque.
4. **popleft():** This function is used to delete an argument from the left end of the deque.
5. **index(ele, beg, end):** This function returns the first index of the value mentioned in arguments, starting searching from beg till end index.
6. **insert(i, a):** This function inserts the value mentioned in arguments(a) at index(i) specified in arguments.
7. **remove():** This function removes the first occurrence of the value mentioned in arguments.
8. **count():** This function counts the number of occurrences of value mentioned in arguments.
9. **len(dequeue):** Return the current size of the dequeue.
10. **Deque[0]:** We can access the front element of the deque using indexing with de[0].
11. **Deque[-1]:** We can access the back element of the deque using indexing with de[-1].
12. **extend(iterable):** This function is used to add multiple values at the right end of the deque. The argument passed is iterable.
13. **extendleft(iterable):** This function is used to add multiple values at the left end of the deque. The argument passed is iterable. Order is reversed as a result of left appends.
14. **reverse():** This function is used to reverse the order of deque elements.
15. **rotate():** This function rotates the deque by the number specified in arguments. If the number specified is negative, rotation occurs to the left. Else rotation is to right.

```

import java.util.ArrayDeque;
import java.util.Deque;

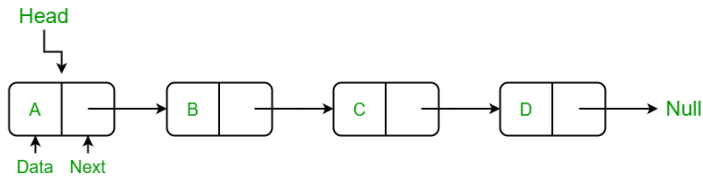
public class DequeOperations
{
    # Write Code Here
}

```

7. Linked List

7.1 Singly Linked List

A singly linked list is a linear data structure in which the elements are not stored in contiguous memory locations and each element is connected only to its next element using a pointer.



Creating a linked list involves the following operations:

1. Creating a Node class:
2. Insertion at beginning:
3. Insertion at end
4. Insertion at middle
5. Update the node
6. Deletion at beginning
7. Deletion at end
8. Deletion at middle
9. Remove last node
10. Linked list traversal
11. Get length

```
class Node {
    String data;
    Node next;

    Node(String data) {
        this.data = data;
        this.next = null;
    }
}

class LinkedList
{
    # Write Code Here
}

public void insertAtEnd(String data)
{
    # Write Code Here
}

public void updateNode(String val, int index)
{
    # Write Code Here
}

public void remove_first_node() {
    # Write Code Here
}

public void remove_last_node()
```

```

    {
        # Write Code Here
    }

    public void remove_at_index(int index)

    {
        # Write Code Here
    }

    public void remove_node(String data)
    {
        # Write Code Here
    }

    public int sizeOfLL()
    {
        # Write Code Here
    }

    public void printLL()
    {
        # Write Code Here
    }

    public static void main(String[] args)
    {
        # Write Code Here
    }
}

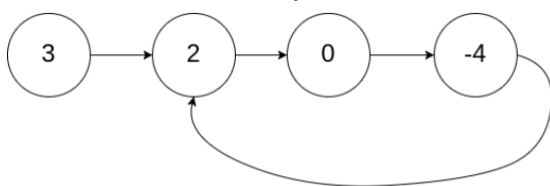
```

7.2 Linked List Cycle

Given head, the head of a linked list, determine if the linked list has a cycle in it. There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the next pointer. Internally, pos is used to denote the index of the node that tail's next pointer is connected to.

Note that pos is not passed as a parameter.

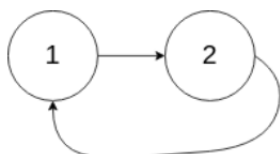
Return true if there is a cycle in the linked list. Otherwise, return false.



Input: head = [3, 2, 0, -4], pos = 1

Output: true

Explanation: There is a cycle in the linked list, where the tail connects to the 1st node (0-indexed).



Input: head = [1, 2], pos = 0

Output: true

Explanation: There is a cycle in the linked list, where the tail connects to the 0th node.

1

Input: head = [1], pos = -1

Output: false

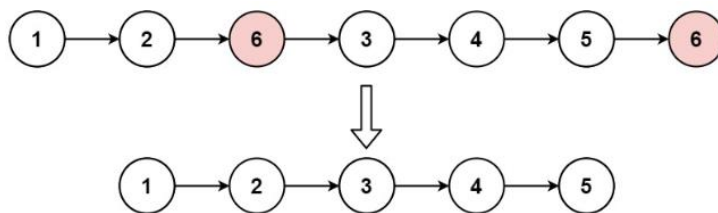
Explanation: There is no cycle in the linked list.

```
class ListNode
{
    # Write Code Here
}

public class Solution
{
    # Write Code Here
}
```

7.3 Remove Linked List Elements

Given the head of a linked list and an integer val, remove all the nodes of the linked list that has Node.val == val, and return the new head.



Input: head = [1, 2, 6, 3, 4, 5, 6], val = 6

Output: [1, 2, 3, 4, 5]

Input: head = [], val = 1

Output: []

Input: head = [7, 7, 7, 7], val = 7

Output: []

```
class ListNode {
    # Write Code Here
}

public class Solution {
    public boolean hasCycle(ListNode head)
    {
        # Write Code Here
    }

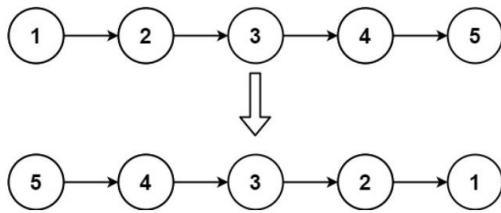
    public static void main(String[] args)
    {
        # Write Code Here
    }
}
```

7.4 Reverse Linked List

Given the head of a singly linked list, reverse the list, and return the reversed list.

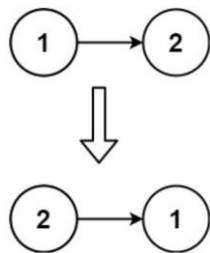
Input: head = [1, 2, 3, 4, 5]

Output: [5, 4, 3, 2, 1]



Input: head = [1, 2]

Output: [2, 1]



```
class ListNode {
    int val;
    ListNode next;

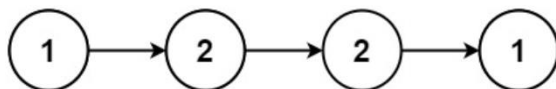
    ListNode(int val) {
        this.val = val;
        this.next = null;
    }
}

public class Solution {
    public ListNode reverseList(ListNode head)
    {
        # Write Code Here
    }

    public static void main(String[] args)
    {
        # Write Code Here
    }
}
```

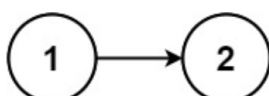
7.5 Palindrome Linked List

Given the head of a singly linked list, return true if it is a palindrome or false otherwise.



Input: head = [1, 2, 2, 1]

Output: true



Input: head = [1, 2]

Output: false

```
class ListNode
{
    # Write Code Here
}
public class Solution {
    public boolean isPalindrome(ListNode head)
    {
        # Write Code Here
    }

    public static void main(String[] args)
    {
        # Write Code Here
    }
}
```

7.6 Middle of the Linked List

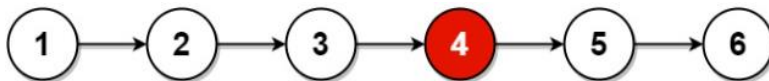
Given the head of a singly linked list, return the middle node of the linked list. If there are two middle nodes, return the second middle node.



Input: head = [1, 2, 3, 4, 5]

Output: [3, 4, 5]

Explanation: The middle node of the list is node 3.



Input: head = [1, 2, 3, 4, 5, 6]

Output: [4, 5, 6]

Explanation: Since the list has two middle nodes with values 3 and 4, we return the second one.

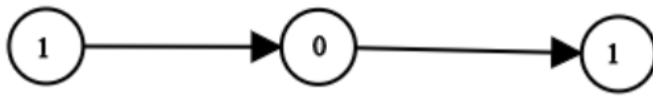
```
class ListNode
{
    # Write Code Here
}
public class Solution
{
    # Write Code Here
}

public static void main(String[] args) {
    # Write Code Here
}
```

7.7 Convert Binary Number in a Linked List to Integer

Given head which is a reference node to a singly-linked list. The value of each node in the linked list is either 0 or 1. The linked list holds the binary representation of a number.

Return the decimal value of the number in the linked list. The most significant bit is at the head of the linked list.



Input: head = [1, 0, 1]

Output: 5

Explanation: (101) in base 2 = (5) in base 10

Input: head = [0]

Output: 0

```
class ListNode {
    int val;
    ListNode next;

    ListNode(int val) {
        this.val = val;
        this.next = null;
    }
}

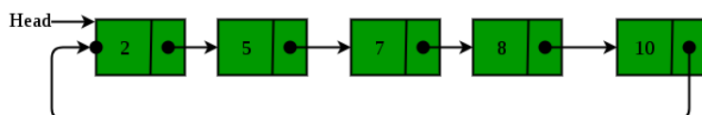
public class Solution
{
    # Write Code Here
}

public static void main(String[] args)
{
    # Write Code Here
}
```

8. Circular Single Linked List and Doubly Linked List

8.1 Circular Linked List

The circular linked list is a linked list where all nodes are connected to form a circle. In a circular linked list, the first node and the last node are connected to each other which forms a circle. There is no NULL at the end.



Operations on the circular linked list:

1. Insertion at the beginning
2. Insertion at the end
3. Insertion in between the nodes
4. Deletion at the beginning
5. Deletion at the end

6. Deletion in between the nodes
7. Traversal

```
import java.util.ArrayList;
public class Main{
    static class Node{
        int data;
        Node next;
        Node(int data){
            this.data = data;
            this.next = null;
        }
    }
    static class CircularLinkedList
    {
        # Write Code Here
    }
    Node addAfter(int data, int item)
    {
        # Write Code Here
    }
    void deleteNode(Node last, int key)
    {
        # Write Code Here
    }
}
```

System.

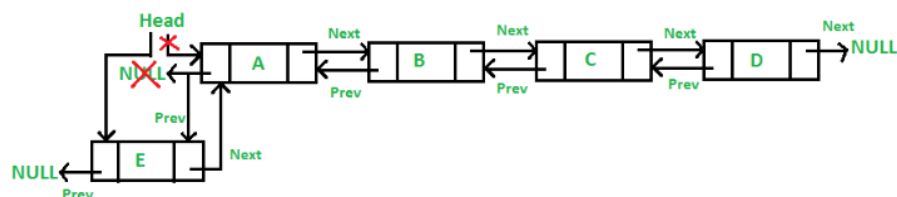
8.2 Doubly Linked List

The A doubly linked list is a type of linked list in which each node consists of 3 components:

1. *prev - address of the previous node
2. data - data item
3. *next - address of next node.



Double Linked List Node



Operations on the Double Linked List:

1. Insertion at the beginning
2. Insertion at the end
3. Insertion in between the nodes
4. Deletion at the beginning
5. Deletion at the end
6. Deletion in between the nodes
7. Traversal


```

import java.util.Scanner;

class Node {
    int data;
    Node next;
    Node prev;

    Node(int data) {
        this.data = data;
        this.next = null;
        this.prev = null;
    }
}

class DLinkedList {
    Node head;
    int ctr;

    DLinkedList() {
        this.head = null;
        this.ctr = 0;
    }

    void insertBeg(int data)
    {
        # Write Code Here
    }

    void insertEnd(int data)
    {
        # Write Code Here
    }

    void deleteBeg()
    {
        # Write Code Here
    }

    void deleteEnd()
    {
        # Write Code Here
    }

    void insertPos(int pos, int data)
    {
        # Write Code Here
    }

    void deletePos(int pos)
    {
        # Write Code Here
    }

    void traverseF()
    {
        # Write Code Here
    }

    void traverseR()

```

```

    {
        # Write Code Here
    }

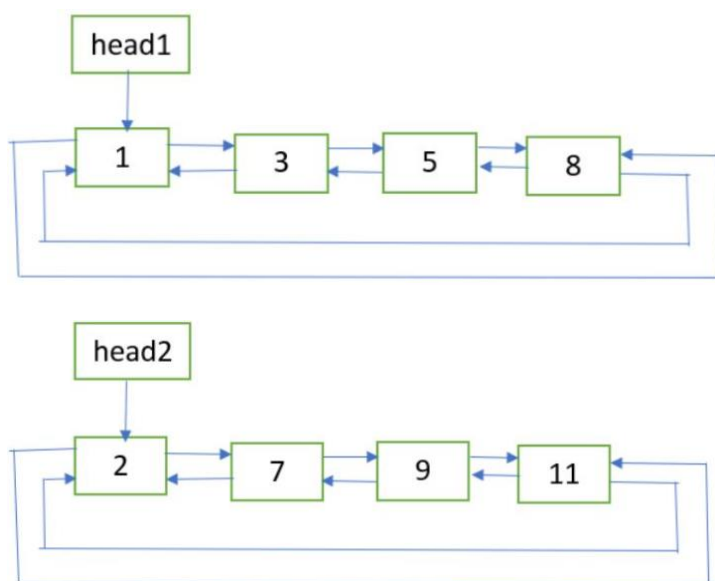
public class Main {
    public static void main(String[] args)
    {
        # Write Code Here
    }
}

```

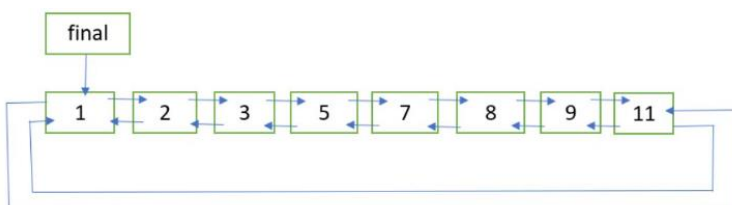
8.3 Sorted Merge of Two Sorted Doubly Circular Linked Lists

Given two sorted Doubly circular Linked List containing n1 and n2 nodes respectively. The problem is to merge the two lists such that resultant list is also in sorted order.

Input: List 1 and List 2



Output: Merged List



Procedure for Merging Doubly Linked List:

1. If head1 == NULL, return head2.
2. If head2 == NULL, return head1.
3. Let **last1** and **last2** be the last nodes of the two lists respectively. They can be obtained with the help of the previous links of the first nodes.
4. Get pointer to the node which will be the last node of the final list. If last1.data < last2.data, then **last_node** = last2, Else **last_node** = last1.
5. Update last1.next = last2.next = NULL.
6. Now merge the two lists as two sorted doubly linked list are being merged.
Refer **merge** procedure of this post. Let the first node of the final list be **finalHead**.

7. Update finalHead.prev = last_node and last_node.next = finalHead.
8. Return **finalHead**.

```

class Node {
    int data;
    Node next, prev;

    Node(int data) {
        this.data = data;
        this.next = null;
        this.prev = null;
    }
}

public class SortedMergeDoublyCircularLinkedList
{
    # Write Code Here
}

static Node mergeUtil(Node head1, Node head2)
{
    # Write Code Here
}

static void printList(Node head)
{
    # Write Code Here
}

public static void main(String[] args)
{
    # Write Code Here
}
}

```

8.4 Delete all occurrences of a given key in a Doubly Linked List

Given a doubly linked list and a key x. The problem is to delete all occurrences of the given key x from the doubly linked list.

Input: 2 <-> 2 <-> 10 <-> 8 <-> 4 <-> 2 <-> 5 <-> 2
x = 2

Output: 10 <-> 8 <-> 4 <-> 5

Algorithm:

delAllOccurOfGivenKey (head_ref, x)

```

if head_ref == NULL
    return
Initialize current = head_ref
Declare next
while current != NULL
    if current->data == x
        next = current->next
        deleteNode(head_ref, current)
        current = next
    else
        current = current->next

```

```

class Node {
    int data;
    Node next, prev;

    Node(int data) {
        this.data = data;
        this.next = null;
        this.prev = null;
    }
}

public class DeleteOccurrenceInDoublyLinkedList
{
    # Write Code Here
}

static Node deleteAllOccurOfX(Node head, int x)
{
    # Write Code Here
}

static void printList(Node head)
{
    # Write Code Here
}

public static void main(String[] args)
{
    # Write Code Here
}
}

```

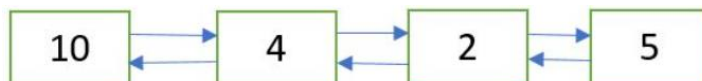
8.5 Delete a Doubly Linked List Node at a Given Position

Given a doubly linked list and a position n. The task is to delete the node at the given position n from the beginning.

Input: Initial doubly linked list



Output: Doubly Linked List after deletion of node at position n = 2



Procedure:

1. Get the pointer to the node at position n by traversing the doubly linked list up to the nth node from the beginning.
2. Delete the node using the pointer obtained in Step 1.

```

class Node {
    int data;
    Node next, prev;

    Node(int data) {

```

```

        this.data = data;
        this.next = null;
        this.prev = null;
    }
}

public class DeleteNodeAtGivenPosition
{
    # Write Code Here
}

static Node deleteNode(Node head, Node del)
{
    # Write Code Here
}

static Node deleteNodeAtGivenPos(Node head, int n)
{
    # Write Code Here
}

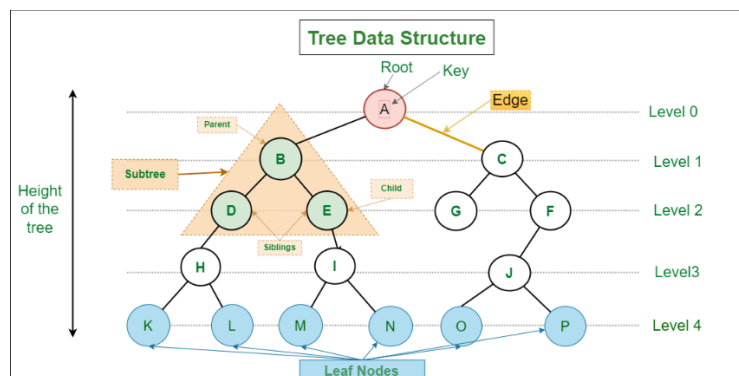
static void printList(Node head)
{
    # Write Code Here
}
}

```

9. Trees

9.1 Tree Creation and Basic Tree Terminologies

A tree data structure is a hierarchical structure that is used to represent and organize data in a way that is easy to navigate and search. It is a collection of nodes that are connected by edges and has a hierarchical relationship between the nodes.



Basic Terminologies in Tree:

1. **Parent Node:** The node which is a predecessor of a node is called the parent node of that node. {B} is the parent node of {D, E}.
2. **Child Node:** The node which is the immediate successor of a node is called the child node of that node. Examples: {D, E} are the child nodes of {B}.
3. **Root Node:** The topmost node of a tree or the node which does not have any parent node is called the root node. {A} is the root node of the tree. A non-empty tree must contain exactly one root node and exactly one path from the root to all other nodes of the tree.
4. **Leaf Node or External Node:** The nodes which do not have any child nodes are called leaf nodes. {K, L, M, N, O, P} are the leaf nodes of the tree.

5. **Ancestor of a Node:** Any predecessor nodes on the path of the root to that node are called Ancestors of that node. {A, B} are the ancestor nodes of the node {E}
6. **Descendant:** Any successor node on the path from the leaf node to that node. {E, I} are the descendants of the node {B}.
7. **Sibling:** Children of the same parent node are called siblings. {D, E} are called siblings.
8. **Level of a node:** The count of edges on the path from the root node to that node. The root node has level 0.
9. **Internal node:** A node with at least one child is called Internal Node.
10. **Neighbour of a Node:** Parent or child nodes of that node are called neighbors of that node.
11. **Subtree:** Any node of the tree along with its descendant.

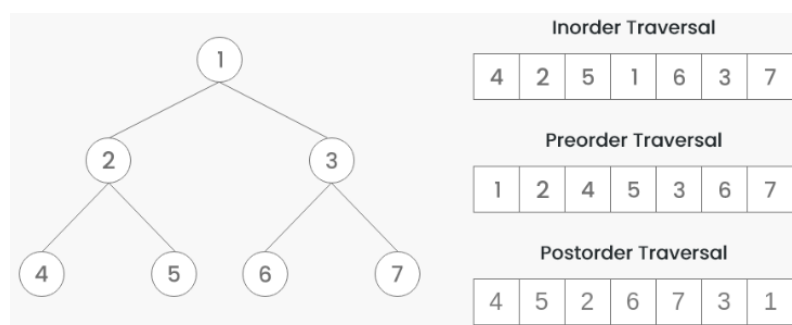
```
import java.util.ArrayList;
import java.util.List;

public class TreeBasicTerminologies
{
    # Write Code Here
}
static void printChildren(int root, List<List<Integer>> adj)
{
    # Write Code Here
}
static void printLeafNodes(int root, List<List<Integer>> adj)
{
    # Write Code Here
}
static void printDegrees(int root, List<List<Integer>> adj)
{
    # Write Code Here
}
public static void main(String[] args)
{
    # Write Code Here
}
}
```

9.2 Binary Tree Traversal Techniques

A binary tree data structure can be traversed in following ways:

1. Inorder Traversal
2. Preorder Traversal
3. Postorder Traversal
4. Level Order Traversal



Algorithm Inorder (tree)

1. Traverse the left subtree, i.e., call Inorder(left->subtree)
2. Visit the root.
3. Traverse the right subtree, i.e., call Inorder(right->subtree)

Algorithm Preorder (tree)

1. Visit the root.
2. Traverse the left subtree, i.e., call Preorder(left->subtree)
3. Traverse the right subtree, i.e., call Preorder(right->subtree)

Algorithm Postorder (tree)

1. Traverse the left subtree, i.e., call Postorder(left->subtree)
2. Traverse the right subtree, i.e., call Postorder(right->subtree)
3. Visit the root.

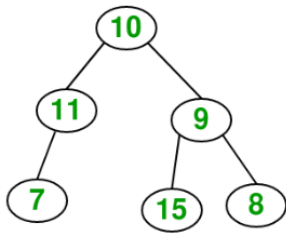
```
import java.util.Scanner;
class Node
{
    # Write Code Here
}
class BT {
    Node root;
    BT() {
        this.root = null;
    }
    void insert(int data)
    {
        # Write Code Here
    }
    Node insertRec(Node root, int data)
    {
        # Write Code Here
    }
    void postorder(Node root)
    {
        # Write Code Here
    }
    void preorder(Node root)
    {
        # Write Code Here
    }
    void inorder(Node root)
    {
        # Write Code Here
    }
}

public class BinaryTreeTraversal {
    public static void main(String[] args)
    {
        # Write Code Here
    }
}
```

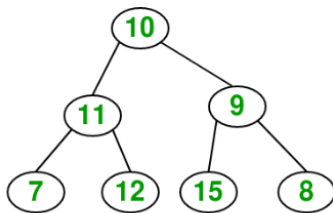
9.3 Insertion in a Binary Tree in Level Order

Given a binary tree and a key, insert the key into the binary tree at the first position available in level order.

Input: Consider the tree given below



Output:



After inserting 12

The idea is to do an iterative level order traversal of the given tree using queue. If we find a node whose left child is empty, we make a new key as the left child of the node. Else if we find a node whose right child is empty, we make the new key as the right child. We keep traversing the tree until we find a node whose either left or right child is empty.

```
class Node
{
    # Write Code Here
}

public class BinaryTreeInsertion
{
    # Write Code Here
    static Node insert(Node root, int key)
    {
        # Write Code Here
    }

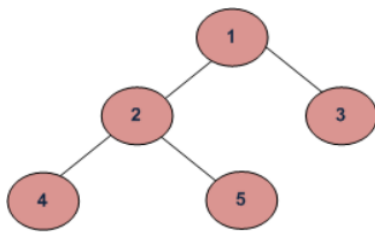
    public static void main(String[] args)
    {
        # Write Code Here
    }
}
```

9.4 Finding the Maximum Height or Depth of a Binary Tree

Given a binary tree, the task is to find the height of the tree. The height of the tree is the number of edges in the tree from the root to the deepest node.

Note: The height of an empty tree is 0.

Input: Consider the tree below



Recursively calculate the height of the left and the right subtrees of a node and assign height to the node as max of the heights of two children plus 1.

$\text{maxDepth}('1') = \max(\text{maxDepth}('2'), \text{maxDepth}('3')) + 1 = 2 + 1$

because recursively

$\text{maxDepth}('2') = \max(\text{maxDepth}('4'), \text{maxDepth}('5')) + 1 = 1 + 1$ and (as height of both '4' and '5' are 1)

$\text{maxDepth}('3') = 1$

Procedure:

- Recursively do a Depth-first search.
- If the tree is empty then return 0
- Otherwise, do the following
 - Get the max depth of the left subtree recursively i.e. call $\text{maxDepth}(\text{tree} \rightarrow \text{left-subtree})$
 - Get the max depth of the right subtree recursively i.e. call $\text{maxDepth}(\text{tree} \rightarrow \text{right-subtree})$
 - Get the max of max depths of left and right subtrees and add 1 to it for the current node.

$$\text{max_depth} = \max(\text{maxdepthofleftsubtree}, \text{maxdepthofrightsubtree}) + 1$$
- Return max_depth.

```

class Node
{
    int data;
    Node left, right;

    Node(int data) {
        this.data = data;
        this.left = null;
        this.right = null;
    }
}

public class MaximumDepthOfTree
{
    # Write Code Here
}

public static void main(String[] args)
{
    # Write Code Here
}

```

9.5 Deletion in a Binary Tree

Given a binary tree, delete a node from it by making sure that the tree shrinks from the bottom (i.e. the deleted node is replaced by the bottom-most and rightmost node).

Input: Delete 10 in below tree

```
    10
   /  \
  20   30
```

Output:

```
    30
   /
  20
```

Input: Delete 20 in below tree

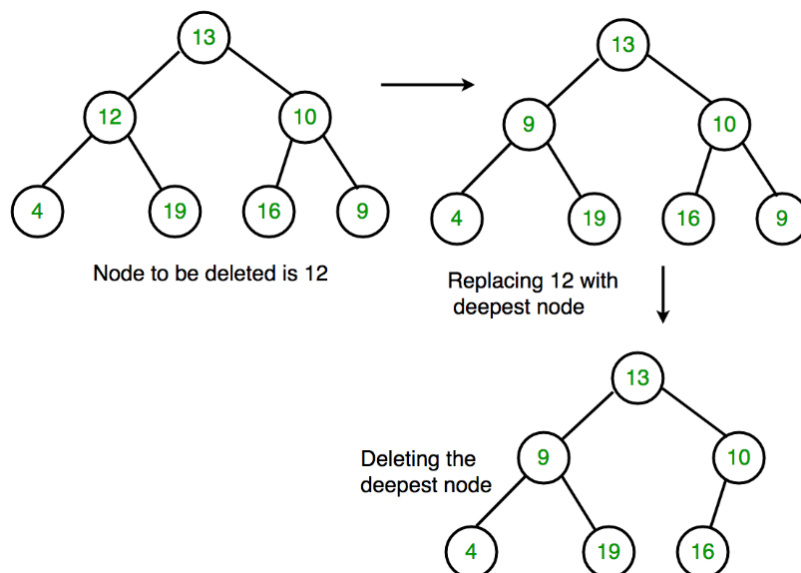
```
    10
   /  \
  20   30
   \
   40
```

Output:

```
    10
   /  \
  40   30
```

Algorithm:

1. Starting at the root, find the deepest and rightmost node in the binary tree and the node which we want to delete.
2. Replace the deepest rightmost node's data with the node to be deleted.
3. Then delete the deepest rightmost node.



```
class Node {
    int data;
    Node left, right;

    Node(int data) {
```

```

        this.data = data;
        this.left = null;
        this.right = null;
    }
}

public class BinaryTreeDeletion
{
    # Write Code Here
}

static void deleteDeepest(Node root, Node dNode)
{
    # Write Code Here
}

static Node deletion(Node root, int key)
{
    # Write Code Here
}

public static void main(String[] args)
{
    # Write Code Here
}
}

```

10. Binary Search Tree (BST)

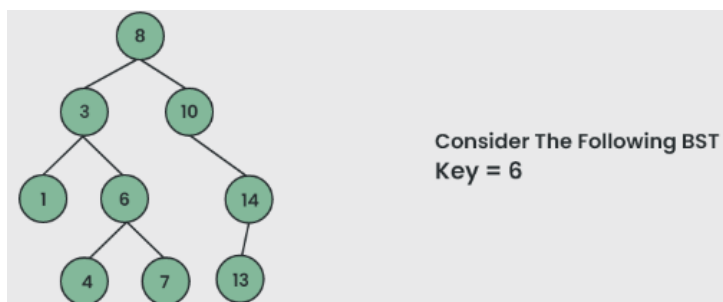
10.1 Searching in Binary Search Tree

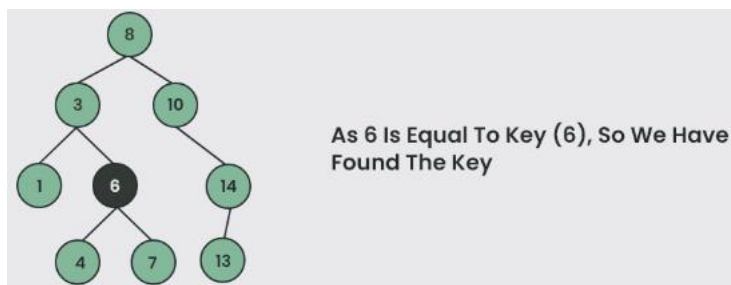
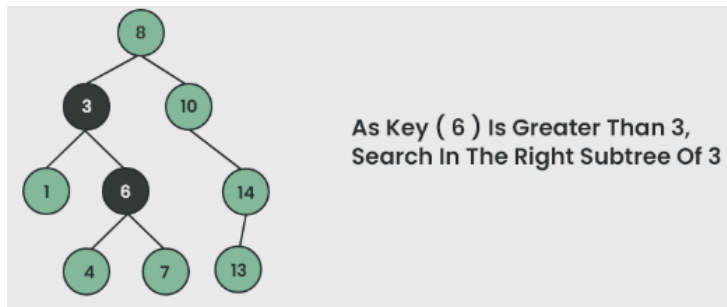
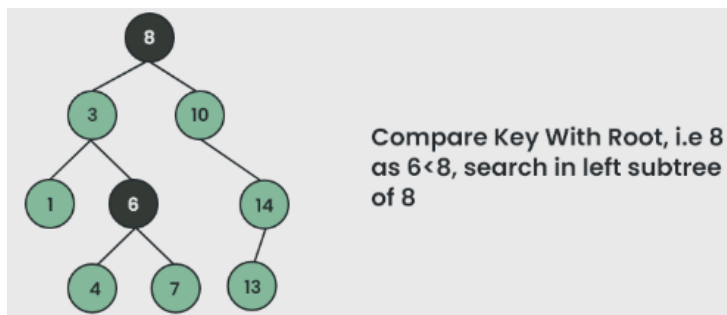
Given a BST, the task is to delete a node in this BST. For searching a value in BST, consider it as a sorted array. Perform search operation in BST using Binary Search Algorithm.

Algorithm to search for a key in a given Binary Search Tree:

Let's say we want to search for the number **X**, We start at the root. Then:

- We compare the value to be searched with the value of the root.
 - If it's equal we are done with the search if it's smaller we know that we need to go to the left subtree because in a binary search tree all the elements in the left subtree are smaller and all the elements in the right subtree are larger.
- Repeat the above step till no more traversal is possible
- If at any iteration, key is found, return True. Else False.





```
// Node class to represent each node of the BST
class Node {
    int key;
    Node left, right;

    public Node(int item) {
        key = item;
        left = right = null;
    }
}

class BST
{
    # Write Code Here
}

Node search(int key) {
    return searchRec(root, key);
}

Node searchRec(Node root, int key)
{
    # Write Code Here
}

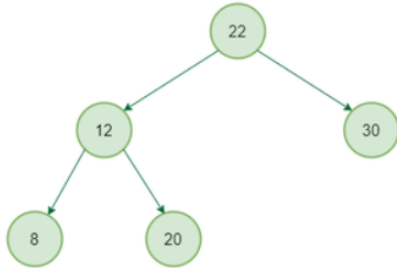
public static void main(String[] args)
{
    # Write Code Here
}
```

```
}
```

10.2 Find the node with Minimum Value in a BST

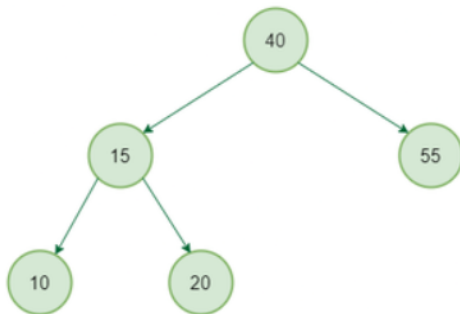
Write a function to find the node with minimum value in a Binary Search Tree.

Input: Consider the tree given below



Output: 8

Input: Consider the tree given below



Output: 10

```
import java.util.ArrayList;
import java.util.List;

class Node {
    int data;
    Node left, right;

    public Node(int item) {
        data = item;
        left = right = null;
    }
}

class BinarySearchTree
{
    # Write Code Here
}

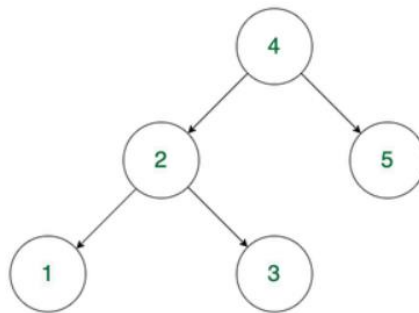
public static void main(String[] args)
{
    # Write Code Here
}
```

10.3 Check if a Binary Tree is BST or not

A binary search tree (BST) is a node-based binary tree data structure that has the following properties.

1. The left subtree of a node contains only nodes with keys less than the node's key.
2. The right subtree of a node contains only nodes with keys greater than the node's key.
3. Both the left and right subtrees must also be binary search trees.
4. Each node (item in the tree) has a distinct key.

Input: Consider the tree given below



Output: Check if max value in left subtree is smaller than the node and min value in right subtree greater than the node, then print it "Is BST" otherwise "Not a BST"

Procedure:

1. If the current node is null then return true
2. If the value of the left child of the node is greater than or equal to the current node then return false
3. If the value of the right child of the node is less than or equal to the current node then return false
4. If the left subtree or the right subtree is not a BST then return false
5. Else return true

```
class Node {
    int data;
    Node left, right;

    public Node(int item) {
        data = item;
        left = right = null;
    }
}

class BinaryTree
{
    # Write Code Here

    boolean isBST(Node node) {
        return isBSTUtil(node, Integer.MIN_VALUE, Integer.MAX_VALUE);
    }

    boolean isBSTUtil(Node node, int min, int max)
    {
        # Write Code Here
    }

    public static void main(String[] args)
```

```

{
  # Write Code Here
}

```

10.4 Second Largest Element in BST

Given a Binary search tree (BST), find the second largest element.

Input: Root of below BST

```

    10
   /
  5

```

Output: 5

Input: Root of below BST

```

    10
   / \
  5  20
     \
     30

```

Output: 20

Procedure: The second largest element is second last element in inorder traversal and second element in reverse inorder traversal. We traverse given Binary Search Tree in reverse inorder and keep track of counts of nodes visited. Once the count becomes 2, we print the node.

```

class Node {
    int key;
    Node left, right;

    public Node(int item) {
        key = item;
        left = right = null;
    }
}

class BinarySearchTree
{
    # Write Code Here

    secondLargestUtil(node.right);
    count++;

    // If count is equal to 2 then this is the second largest
    if (count == 2) {
        System.out.println("The second largest element is " + node.key);
        return;
    }
}

```

```

        secondLargestUtil(node.left);
    }

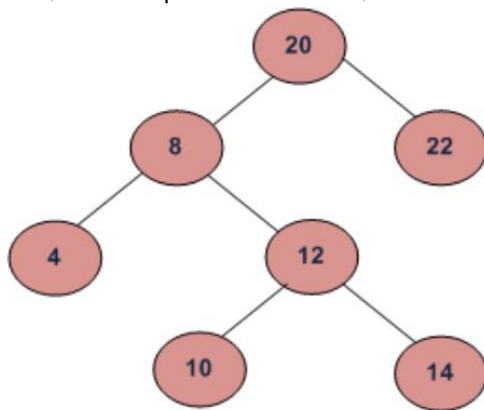
    // Function to find the second largest element
    void secondLargest(Node node) {
        count = 0;
        secondLargestUtil(node);
    }

    // Driver code
    public static void main(String[] args)
    {
        # Write Code Here
    }
}

```

Try:

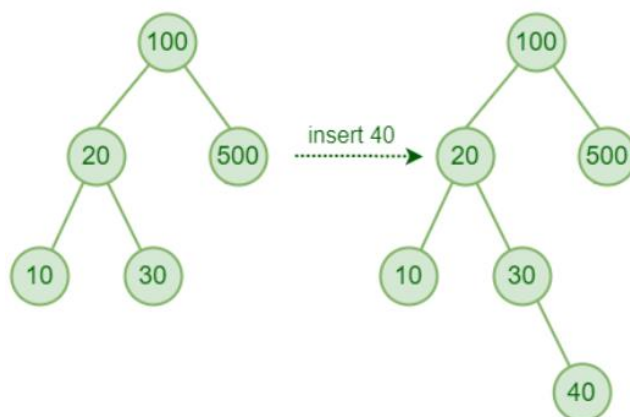
1. **Kth largest element in BST when modification to BST is not allowed:** Given a Binary Search Tree (BST) and a positive integer k, find the k'th largest element in the Binary Search Tree. For a given BST, if k = 3, then output should be 14, and if k = 5, then output should be 10.



10.5 Insertion in Binary Search Tree (BST)

Given a Binary search tree (BST), the task is to insert a new node in this BST.

Input: Consider a BST and insert the element 40 into it.



Procedure for inserting a value in a BST:

A new key is always inserted at the leaf by maintaining the property of the binary search tree. We start searching for a key from the root until we hit a leaf node. Once a leaf node is found, the new node is added as a child of the leaf node. The below steps are followed while we try to insert a node into a binary search tree:

- Check the value to be inserted (say X) with the value of the current node (say val) we are in:
 - If X is less than val move to the left subtree.
 - Otherwise, move to the right subtree.
- Once the leaf node is reached, insert X to its right or left based on the relation between X and the leaf node's value.

```
// A utility class that represents an individual node in a BST
class Node {
    int val;
    Node left, right;

    public Node(int item) {
        val = item;
        left = right = null;
    }
}

class BinarySearchTree
{
    # Write Code Here

}

void inorder()
{
    inorderRec(root);
}

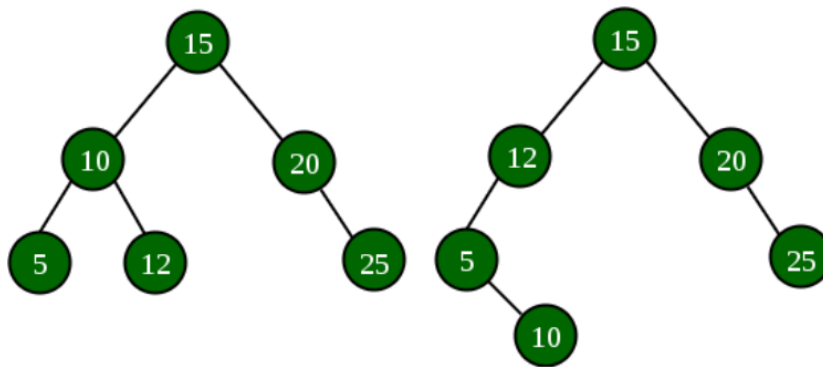
void inorderRec(Node root)
{
    # Write Code Here
}

// Driver code
public static void main(String[] args)
{
    # Write Code Here
}
}
```

Try:

1. **Check if two BSTs contain same set of elements:** Given two Binary Search Trees consisting of unique positive elements, we have to check whether the two BSTs contain the same set of elements or not.

Input: Consider two BSTs which contains same set of elements {5, 10, 12, 15, 20, 25}, but the structure of the two given BSTs can be different.



11. AVL Tree

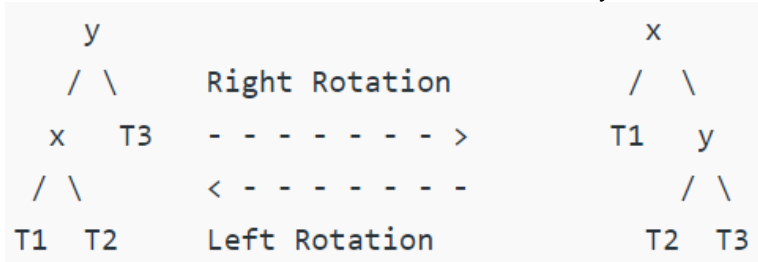
11.1 Insertion in an AVL Tree

AVL tree is a self-balancing Binary Search Tree (BST) where the difference between heights of left and right subtrees cannot be more than one for all nodes. To make sure that the given tree remains AVL after every insertion, we must augment the standard BST insert operation to perform some re-balancing.

Following are two basic operations that can be performed to balance a BST without violating the BST property ($\text{keys}(\text{left}) < \text{key}(\text{root}) < \text{keys}(\text{right})$).

- Left Rotation
- Right Rotation

T1, T2 and T3 are subtrees of the tree, rooted with y (on the left side) or x (on the right side)



Keys in both of the above trees follow the following order

$\text{keys}(T1) < \text{key}(x) < \text{keys}(T2) < \text{key}(y) < \text{keys}(T3)$

So BST property is not violated anywhere.

Procedure for inserting a node into an AVL tree

Let the newly inserted node be w

- Perform standard BST insert for w.
- Starting from w, travel up and find the first unbalanced node. Let z be the first unbalanced node, y be the child of z that comes on the path from w to z and x be the grandchild of z that comes on the path from w to z.
- Re-balance the tree by performing appropriate rotations on the subtree rooted with z. There can be 4 possible cases that need to be handled as x, y and z can be arranged in 4 ways.
- Following are the possible 4 arrangements:
 - y is the left child of z and x is the left child of y (Left Left Case)
 - y is the left child of z and x is the right child of y (Left Right Case)
 - y is the right child of z and x is the right child of y (Right Right Case)
 - y is the right child of z and x is the left child of y (Right Left Case)

```

class TreeNode {
    int val, height;
    TreeNode left, right;

    TreeNode(int d) {
        val = d;
        height = 1;
    }
}
class AVL_Tree {

    # write the code
}

TreeNode leftRotate(TreeNode x)
{
    # write the code
}
public static void main(String[] args) {
    # write the code
}

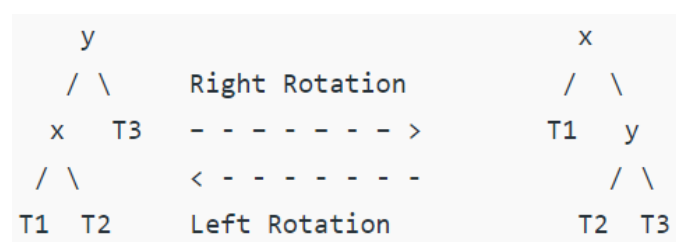
```

11.2 Deletion in an AVL Tree

Given an AVL tree, make sure that the given tree remains AVL after every deletion, we must augment the standard BST delete operation to perform some re-balancing. Following are two basic operations that can be performed to re-balance a BST without violating the BST property ($\text{keys}(\text{left}) < \text{key}(\text{root}) < \text{keys}(\text{right})$).

1. Left Rotation
2. Right Rotation

T1, T2 and T3 are subtrees of the tree rooted with y (on left side) or x (on right side)



Keys in both of the above trees follow the following order

$\text{keys}(T1) < \text{key}(x) < \text{keys}(T2) < \text{key}(y) < \text{keys}(T3)$

So BST property is not violated anywhere.

Procedure to delete a node from AVL tree:

Let w be the node to be deleted

1. Perform standard BST delete for w.

2. Starting from w, travel up and find the first unbalanced node. Let z be the first unbalanced node, y be the larger height child of z, and x be the larger height child of y. Note that the definitions of x and y are different from insertion here.
3. Re-balance the tree by performing appropriate rotations on the subtree rooted with z. There can be 4 possible cases that needs to be handled as x, y and z can be arranged in 4 ways.

Following are the possible 4 arrangements:

- i.y is left child of z and x is left child of y (Left Left Case)
- ii.y is left child of z and x is right child of y (Left Right Case)
- iii.y is right child of z and x is right child of y (Right Right Case)
- iv.y is right child of z and x is left child of y (Right Left Case)

```
class TreeNode
{
    int val, height;
    TreeNode left, right;

    TreeNode(int d) {
        val = d;
        height = 1;
    }
}

class AVL_Tree {

    TreeNode leftRotate(TreeNode z)
    {
        # Write code here
    }

    TreeNode rightRotate(TreeNode z)
    {
        # Write code here
    }

    TreeNode insert(TreeNode node, int key)
    {
        # Write code here
    }
}
```

11.3 Count Greater Nodes in AVL Tree

Given an AVL tree, calculate number of elements which are greater than given value in AVL tree.

Input: x = 5

Root of below AVL tree

```

    9
   / \
  1  10
 / \  \
0  5  11
 / \ \
-1 2  6
```

Output: 4

Explanation: There are 4 values which are greater than 5 in AVL tree which are 6, 9, 10 and 11.

```

class TreeNode {
    int key, height, desc;
    TreeNode left, right;

    TreeNode(int d) {
        key = d;
        height = 1;
        desc = 0;
    }
}

class AVL_Tree
{
    # Write code here

    TreeNode insert(TreeNode node, int key)
    {
        # Write code here
    }
    TreeNode minValueNode(TreeNode node)
    {
        # Write code here
    }

    TreeNode deleteNode(TreeNode root, int key)
    {
        # Write code here
    }
}

void preOrder(TreeNode node)
{
    # Write code here
}

public class Main
{
    # Write code here
}

```

11.4 Minimum Number of Nodes in an AVL Tree with given Height

Given the height of an AVL tree 'h', the task is to find the minimum number of nodes the tree can have.

Input: H = 0

Output: N = 1

Only '1' node is possible if the height of the tree is '0' which is the root node.

Input: H = 3

Output: N = 7

Recursive approach:

In an AVL tree, we have to maintain the height balance property, i.e. difference in the height of the left and the right subtrees cannot be other than -1, 0 or 1 for each node.

We will try to create a recurrence relation to find minimum number of nodes for a given height, $n(h)$.

- For height = 0, we can only have a single node in an AVL tree, i.e. $n(0) = 1$

- For height = 1, we can have a minimum of two nodes in an AVL tree, i.e. $n(1) = 2$
- Now for any height 'h', root will have two subtrees (left and right). Out of which one has to be of height h-1 and other of h-2. [root node excluded]
- So, $n(h) = 1 + n(h-1) + n(h-2)$ is the required recurrence relation for $h \geq 2$ [1 is added for the root node]

```
public class AVLTreeMinimumNodes {

    public static int AVLnodes(int height)
    {
        # Write code here
    }

    public static void main(String[] args) {
        int H = 3;
        System.out.println(AVLnodes(H)); // Output: 4
    }
}
```

12. Graph Traversal

12.1 Breadth First Search

The **Breadth First Search (BFS)** algorithm is used to search a graph data structure for a node that meets a set of criteria. It starts at the root of the graph and visits all nodes at the current depth level before moving on to the nodes at the next depth level.

For a given graph G, print BFS traversal from a given source vertex.

```
import java.util.*;

public class Graph {
    private Map<Integer, List<Integer>> graph;

    public Graph()
    {
        graph = new HashMap<>();
    }

    public void addEdge(int u, int v) {
        if (!graph.containsKey(u)) {
            graph.put(u, new ArrayList<>());
        }
        graph.get(u).add(v);
    }

    public void BFS(int s)
    {
        # Write code here
    }

    public static void main(String[] args)
    {
        # Write code here
    }
}
```

```
}
```

Output: Following is Breadth First Traversal (starting from vertex 2)

2 0 3 1

12.2 Depth First Search

Depth First Traversal (or DFS) for a graph is similar to Depth First Traversal of a tree. The only catch here is, that, unlike trees, graphs may contain cycles (a node may be visited twice). To avoid processing a node more than once, use a boolean visited array. A graph can have more than one DFS traversal.

For a given graph G, print DFS traversal from a given source vertex.

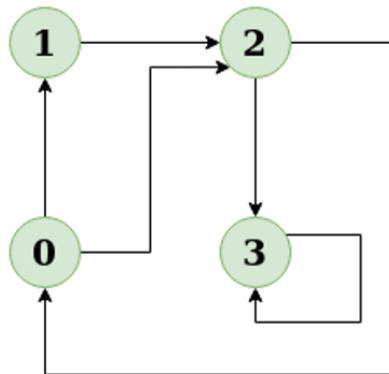
Input: n = 4, e = 6

0 -> 1, 0 -> 2, 1 -> 2, 2 -> 0, 2 -> 3, 3 -> 3

Output: DFS from vertex 1: 1 2 0 3

Explanation:

DFS Diagram:



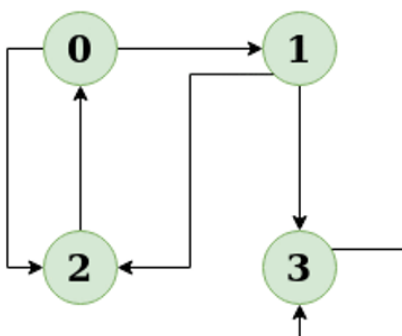
Input: n = 4, e = 6

2 -> 0, 0 -> 2, 1 -> 2, 0 -> 1, 3 -> 3, 1 -> 3

Output: DFS from vertex 2: 2 0 1 3

Explanation:

DFS Diagram:



```
import java.util.*;
class Graph {
    private Map<Integer, List<Integer>> graph;

    public Graph() {
```

```

        // Initialize the graph as a HashMap of ArrayLists
        graph = new HashMap<>();
    }
    public void addEdge(int u, int v)
    {
        # Write code here
    }

    public void DFS(int v) {

        DFSUtil(v, visited);
    }
    public static void main(String[] args)
    {
        # Write code here
    }
}

```

12.3 Best First Search (Informed Search)

The idea of Best First Search is to use an evaluation function to decide which adjacent is most promising and then explore. Best First Search falls under the category of Heuristic Search or Informed Search.

Implementation of Best First Search:

We use a priority queue or heap to store the costs of nodes that have the lowest evaluation function value. So the implementation is a variation of BFS, we just need to change Queue to PriorityQueue.

Algorithm:

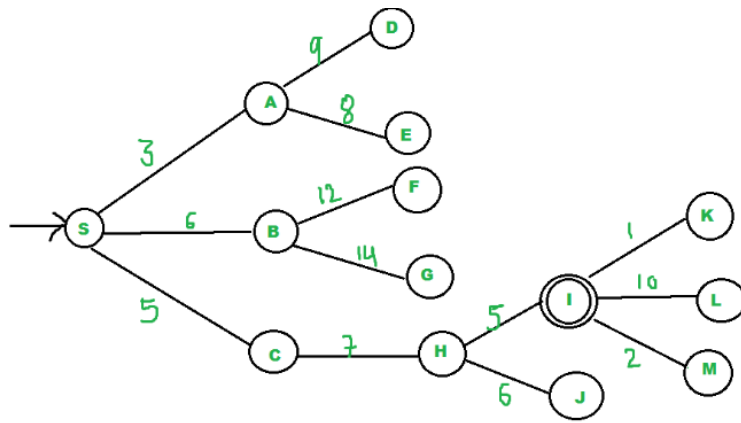
Best-First-Search(Graph g, Node start)

```

1) Create an empty PriorityQueue
   PriorityQueue pq;
2) Insert "start" in pq.
   pq.insert(start)
3) Until PriorityQueue is empty
   u = PriorityQueue.DeleteMin
   If u is the goal
       Exit
   Else
       Foreach neighbor v of u
           If v "Unvisited"
               Mark v "Visited"
               pq.insert(v)
       Mark u "Examined"
End procedure

```

Input: Consider the graph given below.



- We start from source "S" and search for goal "I" using given costs and Best First search.
- pq initially contains S
 - We remove S from pq and process unvisited neighbors of S to pq.
- pq now contains {A, C, B} (C is put before B because C has lesser cost)
- We remove A from pq and process unvisited neighbors of A to pq.
 - pq now contains {C, B, E, D}
- We remove C from pq and process unvisited neighbors of C to pq.
 - pq now contains {B, H, E, D}
- We remove B from pq and process unvisited neighbors of B to pq.
 - pq now contains {H, E, D, F, G}
- We remove H from pq.
- Since our goal "I" is a neighbor of H, we return.

```
import java.util.*;

public class BestFirstSearch {
    static int v = 14;
    static List<List<Pair<Integer, Integer>>> graph = new ArrayList<>();
    static void addedge(int x, int y, int cost) {
        graph.get(x).add(new Pair<>(y, cost));
        graph.get(y).add(new Pair<>(x, cost));
    }

    static void best_first_search(int actual_src, int target, int n)
    {
        # Write code here
    }

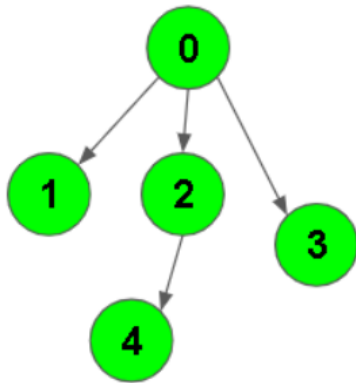
    public static void main(String[] args)
    {
        # Write code here
    }
}
```

12.4 Breadth First Traversal of a Graph

Given a directed graph. The task is to do Breadth First Traversal of this graph starting from 0.

One can move from node u to node v only if there's an edge from u to v. Find the BFS traversal of the graph starting from the 0th vertex, from left to right according to the input graph. Also, you should only take nodes directly or indirectly connected from Node 0 in consideration.

Input: Consider the graph given below where $V = 5$, $E = 4$, edges = $\{(0,1), (0,2), (0,3), (2,4)\}$



Output: 0 1 2 3 4

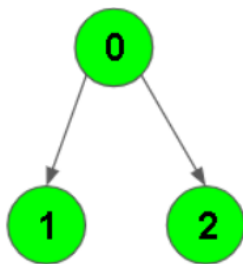
Explanation:

0 is connected to 1, 2, and 3.

2 is connected to 4.

So starting from 0, it will go to 1 then 2 then 3. After this 2 to 4, thus BFS will be 0 1 2 3 4.

Input: Consider the graph given below where $V = 3$, $E = 2$, edges = $\{(0, 1), (0, 2)\}$



Output: 0 1 2

Explanation:

0 is connected to 1, 2. So starting from 0, it will go to 1 then 2, thus BFS will be 0 1 2.

Your task is to complete the function **bfsOfGraph()** which takes the integer V denoting the number of vertices and adjacency list as input parameters and returns a list containing the BFS traversal of the graph starting from the 0th vertex from left to right.

```
import java.util.*;

class Graph {
    private int V;
    private LinkedList<Integer>[] adj;

    Graph(int v) {
        V = v;
        adj = new LinkedList[V];
        for (int i = 0; i < v; ++i)
            adj[i] = new LinkedList();
    }

    void addEdge(int v, int w) {
        adj[v].add(w);
    }
}
```

```

void BFS(int s)
{
    # Write Code Here
}

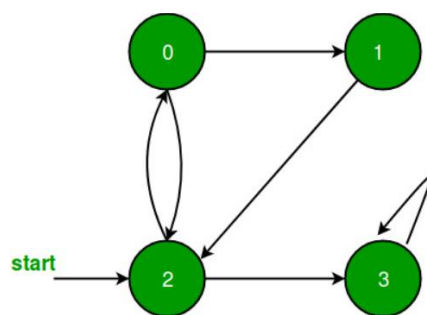
public static void main(String args[]) {
    # Write Code Here
}

```

12.5 Depth First Search (DFS) for Disconnected Graph

Given a Disconnected Graph, the task is to implement DFS or Depth First Search Algorithm for this Disconnected Graph.

Input: Consider the graph given below.



Output: 0 1 2 3

Procedure for DFS on Disconnected Graph:

Iterate over all the vertices of the graph and for any unvisited vertex, run a DFS from that vertex.

```

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

// Class representing a directed graph using adjacency list representation
class Graph
{
    # Write Code Here
}

public Graph()
{
    graph = new HashMap<>();
}

public void addEdge(int u, int v)
{
    # Write Code Here
}

private void DFSUtil(int v, boolean[] visited)
{
    # Write Code Here
}
}

```

```

public void DFS() {
    boolean[] visited = new boolean[graph.size()];
    # Write Code Here
}

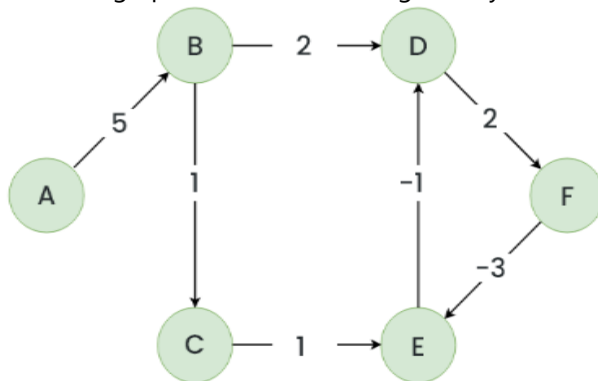
public static void main(String[] args)
{
    # Write Code Here
}
}

```

Try:

1. **Detect a negative cycle in a Graph (Bellman Ford):** A Bellman-Ford algorithm is also guaranteed to find the shortest path in a graph, similar to Dijkstra's algorithm. Although Bellman-Ford is slower than Dijkstra's algorithm, it is capable of handling graphs with negative edge weights, which makes it more versatile. The shortest path cannot be found if there exists a negative cycle in the graph. If we continue to go around the negative cycle an infinite number of times, then the cost of the path will continue to decrease (even though the length of the path is increasing).

Consider a graph G and detect a negative cycle in the graph using Bellman Ford algorithm.



13. Minimum Spanning Tree (MST)

13.1 Kruskal's Algorithm

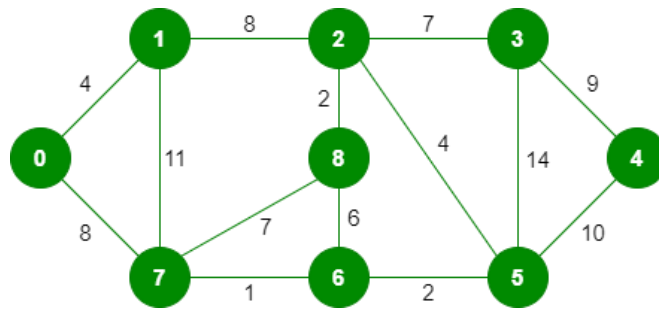
In Kruskal's algorithm, sort all edges of the given graph in increasing order. Then it keeps on adding new edges and nodes in the MST if the newly added edge does not form a cycle. It picks the minimum weighted edge at first and the maximum weighted edge at last.

MST using Kruskal's algorithm:

4. Sort all the edges in non-decreasing order of their weight.
5. Pick the smallest edge. Check if it forms a cycle with the spanning tree formed so far. If the cycle is not formed, include this edge. Else, discard it.
6. Repeat step#2 until there are (V-1) edges in the spanning tree.

Kruskal's algorithm to find the minimum cost spanning tree uses the greedy approach. The Greedy Choice is to pick the smallest weight edge that does not cause a cycle in the MST constructed so far.

Input: For the given graph G find the minimum cost spanning tree.



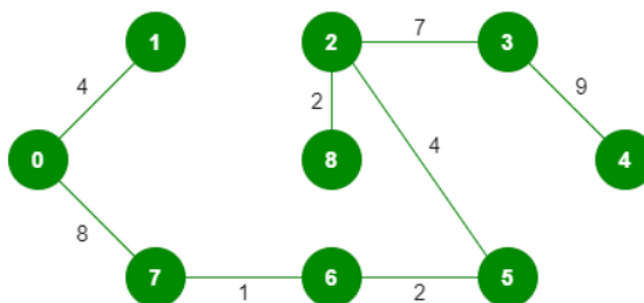
The graph contains 9 vertices and 14 edges. So, the minimum spanning tree formed will be having $(9 - 1) = 8$ edges.

After sorting:

Weight	Source	Destination
1	7	6
2	8	2
2	6	5
4	0	1
4	2	5
6	8	6
7	2	3
7	7	8
8	0	7
8	1	2
9	3	4
10	5	4
11	1	7
14	3	5

Now pick all edges one by one from the sorted list of edges.

Output:



```
// Kruskal's algorithm to find minimum Spanning Tree of a given connected,
import java.util.*;
```

```
public class Graph {
```

```
    class Edge implements Comparable<Edge> {
        int src, dest, weight;
```

```

        // Comparator function used for sorting edges
        public int compareTo(Edge compareEdge) {
            return this.weight - compareEdge.weight;
        }
    }

    private int V; // Number of vertices
    private List<Edge> edges; // List of edges

    public Graph(int vertices) {
        this.V = vertices;
        this.edges = new ArrayList<>();
    }

    public void addEdge(int u, int v, int w)
    {
        # Write Code Here
    }

    private void union(int[] parent, int[] rank, int x, int y)
    {
        # Write Code Here
    }

    public void KruskalMST()
    {
        # Write Code Here
    }

    public static void main(String[] args) {
        Graph g = new Graph(4);
        g.addEdge(0, 1, 10);
        g.addEdge(0, 2, 6);
        g.addEdge(0, 3, 5);
        g.addEdge(1, 3, 15);
        g.addEdge(2, 3, 4);

        // Function call
        g.KruskalMST();
    }
}

```

Output: Following are the edges in the constructed MST

2 -- 3 == 4

0 -- 3 == 5

0 -- 1 == 10

Minimum Cost Spanning Tree: 19

13.2 Prim's Algorithm

The Prim's algorithm starts with an empty spanning tree. The idea is to maintain two sets of vertices. The first set contains the vertices already included in the MST, and the other set contains the vertices not yet included. At every step, it considers all the edges that connect the two sets and picks the minimum weight edge from these edges. After picking the edge, it moves the other endpoint of the edge to the set containing MST.

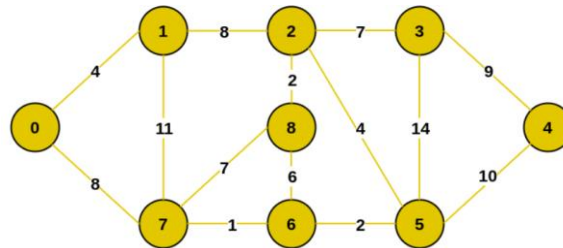
Prim's Algorithm:

The working of Prim's algorithm can be described by using the following steps:

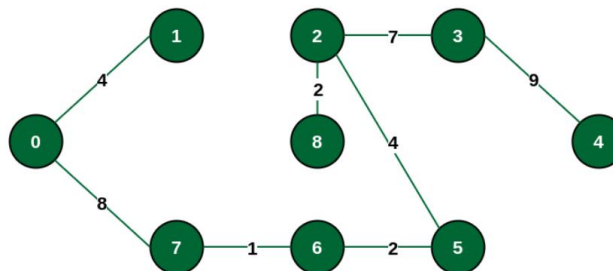
7. Determine an arbitrary vertex as the starting vertex of the MST.

8. Follow steps 3 to 5 till there are vertices that are not included in the MST (known as fringe vertex).
9. Find edges connecting any tree vertex with the fringe vertices.
10. Find the minimum among these edges.
11. Add the chosen edge to the MST if it does not form any cycle.
12. Return the MST and exit

Input: For the given graph G find the minimum cost spanning tree.



Output: The final structure of the MST is as follows and the weight of the edges of the MST is $(4 + 8 + 1 + 2 + 4 + 2 + 7 + 9) = 37$.



```
import java.util.Arrays;

public class Graph {

    private int V; // Number of vertices
    private int[][] graph; // Adjacency matrix representation of graph

    public Graph(int vertices) {
        this.V = vertices;
        this.graph = new int[V][V];
    }

    public void printMST(int[] parent) {
        System.out.println("Edge \tWeight");
        for (int i = 1; i < V; i++) {
            System.out.println(parent[i] + " - " + i + "\t" +
graph[i][parent[i]]);
        }
    }

    private int minKey(int[] key, boolean[] mstSet)
    {
        # Write Code Here
    }

    public void primMST()
    {
        # Write Code Here
    }

    public static void main(String[] args)
```

```

{
    # Write Code Here
}

```

Output:

Edge	Weight
0 - 1	2
1 - 2	3
0 - 3	6
1 - 4	5

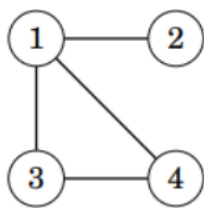
13.3 Total Number of Spanning Trees in a Graph

If a graph is a complete graph with n vertices, then total number of spanning trees is $n^{(n-2)}$ where n is the number of nodes in the graph. In complete graph, the task is equal to counting different labeled trees with n nodes for which have Cayley's formula.

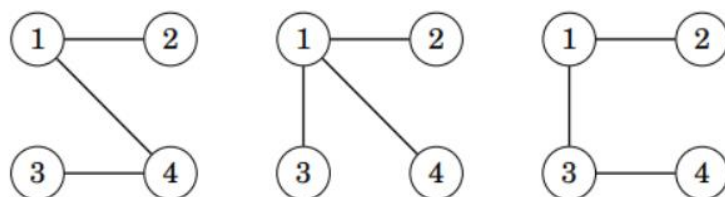
Laplacian matrix:

A Laplacian matrix L , where $L[i, i]$ is the degree of node i and $L[i, j] = -1$ if there is an edge between nodes i and j , and otherwise $L[i, j] = 0$.

Kirchhoff's theorem provides a way to calculate the number of spanning trees for a given graph as a determinant of a special matrix. Consider the following graph,



All possible spanning trees are as follows:



In order to calculate the number of spanning trees, construct a Laplacian matrix L , where $L[i, i]$ is the degree of node i and $L[i, j] = -1$ if there is an edge between nodes i and j , and otherwise $L[i, j] = 0$. for the above graph, The Laplacian matrix will look like this

$$L = \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 2 & -1 \\ -1 & 0 & -1 & 2 \end{bmatrix}$$

The number of spanning trees equals the determinant of a matrix.

The Determinant of a matrix that can be obtained when we remove any row and any column from L . For example, if we remove the first row and column, the result will be,

$$\det \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix} = 3.$$

The determinant is always the same, regardless of which row and column we remove from L.

```
import java.util.Arrays;

public class NumberOfSpanningTrees {

    static final int MAX = 100;
    static final int MOD = 1000000007;
    void multiply(long[][] A, long[][] B, long[][] C, int size) {
        for (int i = 0; i < size; i++) {
            for (int j = 0; j < size; j++) {
                C[i][j] = 0;
                for (int k = 0; k < size; k++) {
                    C[i][j] = (C[i][j] + A[i][k] * B[k][j]) % MOD;
                }
            }
        }
    }
    void power(long[][] A, int N, long[][] result, int size)
    {
        # Write Code Here
    }

    long numOfSpanningTree(int[][] graph, int V)
    {
        # Write Code Here
    }

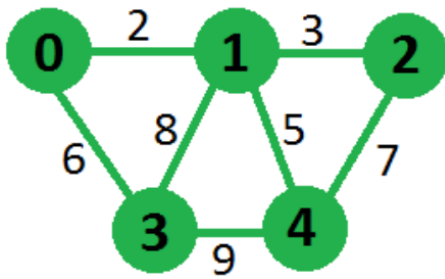
    public static void main(String[] args) {
        int V = 4; // Number of vertices in graph
        int E = 5; // Number of edges in graph
        int[][] graph = { { 0, 1, 1, 1 }, { 1, 0, 1, 1 }, { 1, 1, 0, 1 }, { 1, 1,
1, 0 } };

        NumberOfSpanningTrees obj = new NumberOfSpanningTrees();
        System.out.println(obj.numOfSpanningTree(graph, V));
    }
}
```

13.4 Minimum Product Spanning Tree

A minimum product spanning tree for a weighted, connected, and undirected graph is a spanning tree with a weight product less than or equal to the weight product of every other spanning tree. The weight product of a spanning tree is the product of weights corresponding to each edge of the spanning tree. All weights of the given graph will be positive for simplicity.

Input:



Output: Minimum Product that we can obtain is 180 for above graph by choosing edges 0-1, 1-2, 0-3 and 1-4

This problem can be solved using standard minimum spanning tree algorithms like Kruskal and prim's algorithm, but we need to modify our graph to use these algorithms. Minimum spanning tree algorithms tries to minimize the total sum of weights, here we need to minimize the total product of weights. We can use the property of logarithms to overcome this problem.

$$\log(w_1 * w_2 * w_3 * \dots * w_N) = \log(w_1) + \log(w_2) + \log(w_3) + \dots + \log(w_N)$$

We can replace each weight of the graph by its log value, then we apply any minimum spanning tree algorithm which will try to minimize the sum of $\log(w_i)$ which in turn minimizes the weight product.

```
import java.util.Arrays;
```

```

public class MinimumProductMST {
    // Number of vertices in the graph
    static final int V = 5;

    // A utility function to find the vertex with minimum key value, from the set
of
    // vertices not yet included in MST
    int minKey(int key[], boolean mstSet[]) {
        int min = Integer.MAX_VALUE, min_index = -1;

        for (int v = 0; v < V; v++) {
            if (mstSet[v] == false && key[v] < min) {
                min = key[v];
                min_index = v;
            }
        }

        return min_index;
    }

    void printMST(int parent[], int n, int graph[][])
    {
        # Write Code Here
    }
    void primMST(int inputGraph[][], int logGraph[][])
    {
        # Write Code Here
    }
    void minimumProductMST(int graph[][])
    {
        # Write Code Here
    }
    public static void main(String[] args)
    {
        # Write Code Here
    }
}

```

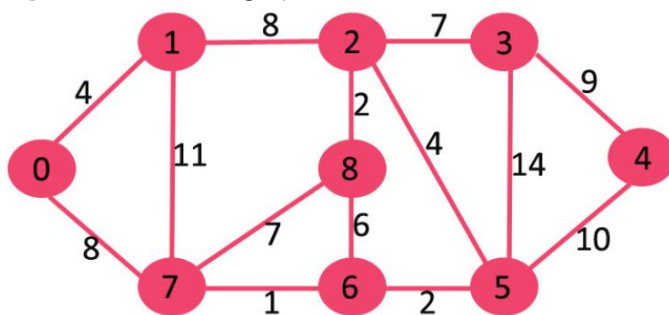
13.5 Reverse Delete Algorithm for Minimum Spanning Tree

In Reverse Delete algorithm, we sort all edges in decreasing order of their weights. After sorting, we one by one pick edges in decreasing order. We include current picked edge if excluding current edge causes disconnection in current graph. The main idea is delete edge if its deletion does not lead to disconnection of graph.

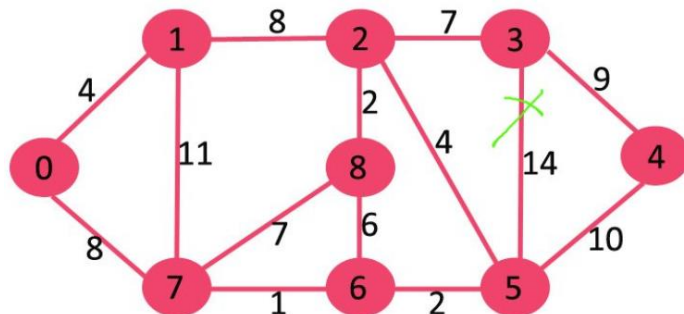
Algorithm:

1. Sort all edges of graph in non-increasing order of edge weights.
2. Initialize MST as original graph and remove extra edges using step 3.
3. Pick highest weight edge from remaining edges and check if deleting the edge disconnects the graph or not.
If disconnects, then we don't delete the edge.
Else we delete the edge and continue.

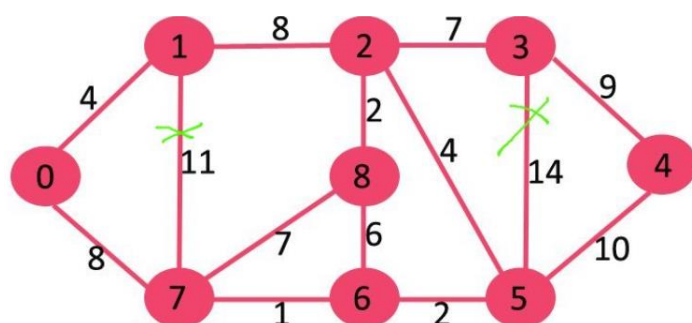
Input: Consider the graph below



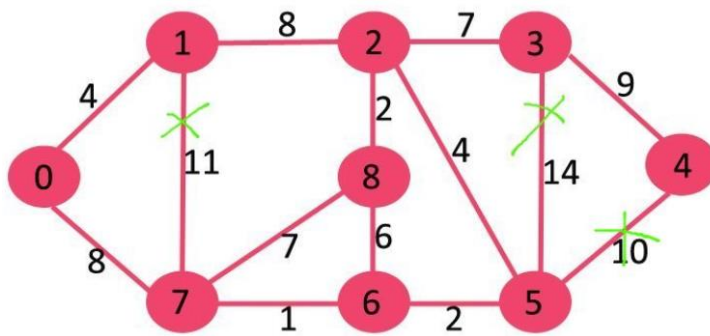
If we delete highest weight edge of weight 14, graph doesn't become disconnected, so we remove it.



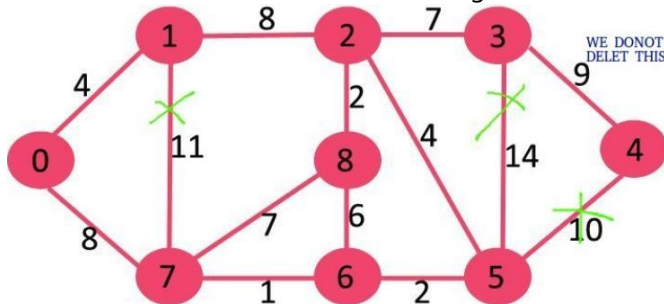
Next we delete 11 as deleting it doesn't disconnect the graph.



Next we delete 10 as deleting it doesn't disconnect the graph.



Next is 9. We cannot delete 9 as deleting it causes disconnection.



We continue this way and following edges remain in final MST.

Edges in MST

(3, 4)
(0, 7)
(2, 3)
(2, 5)
(0, 1)
(5, 6)
(2, 8)
(6, 7)

```
import java.util.ArrayList;
import java.util.Collections;

// Edge class to represent edges in the graph
class Edge {
    int src, dest, weight;

    Edge(int src, int dest, int weight) {
        this.src = src;
        this.dest = dest;
        this.weight = weight;
    }
}

class Graph
{
    # Write Code Here
}

// Function to add an edge to the graph
void addEdge(int u, int v, int w) {
    # Write Code Here
}
```

```

    }
    void dfs(int v, boolean[] visited)
    {
        # Write Code Here
    }

    // Function to check if the graph is connected
    boolean connected()
    {
        # Write Code Here
    }
    void reverseDeleteMST()
    {
        # Write Code Here
    }
}

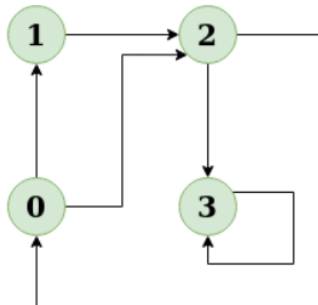
public class ReverseDeleteMST {
    public static void main(String[] args)
    {
        # Write Code Here
    }
}

```

Try:

1. **Detect Cycle in a Directed Graph:** Given the root of a Directed graph, The task is to check whether the graph contains a cycle or not.

Input: N = 4, E = 6



Output: Yes

Explanation: The diagram clearly shows a cycle 0 -> 2 -> 0

14. Final Notes

The only way to learn programming is program, program and program on challenging problems. The problems in this tutorial are certainly NOT challenging. There are tens of thousands of challenging problems available – used in training for various programming contests (such as International Collegiate Programming Contest (ICPC), International Olympiad in Informatics (IOI)). Check out these sites:

- The ACM - ICPC International collegiate programming contest (<https://icpc.global/>)
- The Topcoder Open (TCO) annual programming and design contest (<https://www.topcoder.com/>)
- Universidad de Valladolid's online judge (<https://uva.onlinejudge.org/>).
- Peking University's online judge (<http://poj.org/>).

- USA Computing Olympiad (USACO) Training Program @ <http://train.usaco.org/usacogate>.
- Google's coding competitions (<https://codingcompetitions.withgoogle.com/codejam>, <https://codingcompetitions.withgoogle.com/hashcode>)
- The ICFP programming contest (<https://www.icfpconference.org/>)
- BME International 24-hours programming contest (<https://www.challenge24.org/>)
- The International Obfuscated C Code Contest (<https://www0.us.ioccc.org/main.html>)
- Internet Problem Solving Contest (<https://ipsc.ksp.sk/>)
- Microsoft Imagine Cup (<https://imaginecup.microsoft.com/en-us>)
- Hewlett Packard Enterprise (HPE) Codewars (<https://hpecodewars.org/>)
- OpenChallenge (<https://www.openchallenge.org/>)

Coding Contests Scores

Students must solve problems and attain scores in the following coding contests:

	Name of the contest	Minimum number of problems to solve	Required score
•	CodeChef	20	200
•	Leetcode	20	200
•	GeeksforGeeks	20	200
•	SPOJ	5	50
•	InterviewBit	10	1000
•	Hackerrank	25	250
•	Codeforces	10	100
•	BuildIT	50	500
	Total score need to obtain		2500

Student must have any one of the following certification:

1. HackerRank - Problem Solving Skills Certification (Basic and Intermediate)
2. GeeksforGeeks – Data Structures and Algorithms Certification
3. CodeChef - Learn Data Structures and Algorithms Certification
4. Interviewbit – DSA pro / Python pro
5. Edx – Data Structures and Algorithms
6. NPTEL – Programming, Data Structures and Algorithms
7. NPTEL – Introduction to Data Structures and Algorithms
8. NPTEL – Data Structures and Algorithms
9. NPTEL – Programming and Data Structure

V. TEXT BOOKS:

1. Rance D. Necaise, "Data Structures and Algorithms using Python", Wiley Student Edition.
2. Benjamin Baka, David Julian, "Python Data Structures and Algorithms", Packt Publishers, 2017.

VI. REFERENCE BOOKS:

1. S. Lipschutz, "Data Structures", Tata McGraw Hill Education, 1st edition, 2008.
2. D. Samanta, "Classic Data Structures", PHI Learning, 2nd edition, 2004.

VII. ELECTRONICS RESOURCES:

1. https://www.tutorialspoint.com/data_structures_algorithms/algorithms_basics.htm
2. <https://www.codechef.com/certification/data-structures-and-algorithms/prepare>
3. <https://www.cs.auckland.ac.nz/software/AlgAnim/dsToC.html>
4. <https://online-learning.harvard.edu/course/data-structures-and-algorithms>

VIII. MATERIALS ONLINE

1. Course Content
2. Laboratory lab manual

EXPERIENTIAL ENGINEERING EDUCATION (EXEED) – PROTOTYPE / DESIGN BUILDING

III Semester: AE / ME / CE / ECE / EEE / CSE / CSE (AI & ML) / CSE (DS) / CSE (CS) / IT								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
ACSD12	Skill	L	T	P	C	CIA	SEE	Total
		0	0	2	1	40	60	100
Contact Classes: NIL	Tutorial Classes: NIL	Practical Classes:45				Total Classes: 45		
Prerequisite: Essentials of Innovation.								

I. COURSE OVERVIEW:

This course provides an overall exposure to the various methods and tools of prototyping. This course discusses Low- Fidelity, paper, wireframing and tool-based prototyping techniques along with design principles and patterns.

II. COURSES OBJECTIVES:

The students will try to learn

- I. The basic principles and design aspect of prototyping.
- II. The various techniques, design guidelines and patterns.
- III. The applications of prototyping using various tools and platforms.

III. COURSE CONTENT:

Module 1: An Introduction to Prototyping

Objective: Understand the basics of prototyping and its importance in the design process.

Exercise:

1. Introduction Lecture (10 minutes): Overview of prototyping and its types.
2. Brainstorming Session (20 minutes): Brainstorm simple app ideas.
3. Sketching Session (20 minutes): Create initial design sketches on paper.
4. Paper Prototyping (40 minutes): Build a low-fidelity paper prototype of the app's user interface.
5. Presentation and Feedback (20 minutes): Present prototypes and receive feedback.

Materials and Tools: Paper, pencils, rulers, scissors, glue.

Module 2: Low-Fidelity Prototyping and Paper Prototyping

Objective: Learn concepts and techniques of low-fidelity and paper prototyping.

Exercise:

1. Lecture (10 minutes): Concepts of low-fidelity prototyping.
2. Hands-On Session (40 minutes): Build paper prototypes of the app's screens and user flow.
3. Peer Review (30 minutes): Exchange prototypes with peers for feedback.
4. Iteration (30 minutes): Refine prototypes based on feedback.

Materials and Tools: Paper, pencils, rulers, scissors, glue.

Module 3: Wireframing and Tool-Based Prototyping

Objective: Create wireframes and digital prototypes using software tools.

Exercise:

1. Lecture (10 minutes): Introduction to wireframing and digital prototyping tools.
2. Wireframing (30 minutes): Design wireframes of the app's user interface using tools like Figma or Sketch.
3. Tool-Based Prototyping (50 minutes): Create a digital prototype of the app.
4. Presentation and Feedback (20 minutes): Share digital prototypes and gather feedback.

Materials and Tools: Wireframing tools, computers.

Module 4: Physical Low-Fidelity Prototyping and 3D Printing

Objective: Build simple physical prototypes using basic materials and introduce 3D printing.

Exercise:

1. Lecture (10 minutes): Importance of physical prototyping.
2. Prototype Building (30 minutes): Create a physical representation of the app's main interface using materials like cardboard and foam.
3. **Introduction to 3D Printing (20 minutes):** Overview of 3D printing technology and its applications in prototyping.
4. Testing (20 minutes): Test the physical prototypes for basic functionality.
5. Feedback Session (30 minutes): Present and discuss prototypes.

Materials and Tools: Cardboard, foam, clay, scissors, glue, 3D printer.

Module 5: Tool-Based Prototyping and Circuit Design

Objective: Develop advanced digital prototypes using specialized tools and introduce basic circuit design concepts.

Exercise:

1. Lecture (10 minutes): Advanced prototyping tools and techniques, introduction to circuit design.
2. Hands-On Session (40 minutes): Use tools like Figma, Adobe XD, or Android Studio to create an interactive prototype of the app.
3. **Circuit Design (20 minutes):** Design simple circuits that could be integrated into the app's hardware prototype.
4. User Testing (20 minutes): Conduct user tests and document feedback.
5. Feedback and Iteration (20 minutes): Refine prototypes based on user feedback.

Materials and Tools: Advanced prototyping software, computers, circuit design tools (e.g., breadboards, resistors, LEDs).

Module 6: Design Principles and Patterns - Graphic Design

Objective: Apply graphic design principles to create visually appealing app interfaces.

Exercise:

1. Lecture (10 minutes): Graphic design principles (contrast, alignment, repetition, proximity).
2. Design Session (50 minutes): Create a graphic design layout for the app's interface.
3. Peer Review (30 minutes): Exchange designs for feedback.
4. Refinement (20 minutes): Refine designs based on peer feedback.

Materials and Tools: Graphic design software, computers.

Module 7: Interaction Design and Arduino Integration

Objective: Apply interaction design principles to create intuitive app interfaces and introduce Arduino for hardware interactions.

Exercise:

1. Lecture (10 minutes): Basics of interaction design and Arduino integration.
2. Design Session (40 minutes): Develop interactive wireframes or prototypes for the app's user interactions.
3. **Arduino Integration (20 minutes):** Connect simple Arduino circuits to the app prototype to simulate interactions (e.g., button presses, LED responses).
4. User Testing (20 minutes): Conduct usability tests with peers.
5. Iteration (20 minutes): Improve designs based on test results.

Materials and Tools: Interaction design tools, computers, Arduino kits.

Module 8: Commercial Design Guidelines and Standards

Objective: Design according to industry standards and guidelines.

Exercise:

1. Lecture (10 minutes): Overview of commercial design guidelines
2. Design Session (50 minutes): Develop a prototype of the app following specific design guidelines.
3. Review (30 minutes): Evaluate prototypes against the guidelines.
4. Presentation and Discussion (20 minutes): Present prototypes and discuss adherence to guidelines.

Materials and Tools: Design guidelines documents, design software, computers.

Module 9: Universal Design: Sensory and Cognitive Impairments

Objective: Create app designs accessible to users with sensory and cognitive impairments.

Exercise:

1. Lecture (10 minutes): Principles of universal design.
2. Design Session (50 minutes): Develop an accessible design for the app.
3. User Testing (20 minutes): Test designs with peers simulating impairments.
4. Feedback Session (30 minutes): Present and discuss accessibility features.

Materials and Tools: Design software, computers, accessibility guidelines.

Module 10: Universal Design: Tools, Limitations, and Standards

Objective: Understand tools and limitations in universal design, and adhere to standards.

Exercise:

1. Lecture (10 minutes): Tools and limitations in universal design.
2. Hands-On Session (50 minutes): Use tools to create universally designed app prototypes.
3. Testing (20 minutes): Evaluate prototypes for accessibility and adherence to standards.
4. Iteration (30 minutes): Refine designs based on testing outcomes.

Materials and Tools: Universal design tools, computers.

Module 11: Mobile UI Design, CNC, and Arduino

Objective: Design user interfaces for mobile and wearable devices, create related hardware prototypes using CNC tools, and integrate Arduino for interactions.

Exercise:

1. Lecture (10 minutes): Principles of mobile UI and wearable design, introduction to CNC lathe and mill, and Arduino for wearable interactions.
2. Design Session (30 minutes): Develop a UI prototype for a mobile app or wearable device.
3. **CNC Prototyping (20 minutes):** Create physical components of the wearable device using CNC lathe and mill.
4. **Arduino Integration (20 minutes):** Program Arduino for basic interactions (e.g., motion detection, feedback via LEDs).
5. User Testing (20 minutes): Conduct usability tests on the prototype.
6. Feedback Session (30 minutes): Present and discuss usability findings.

Materials and Tools: Mobile UI design tools, computers, mobile devices, CNC lathe, CNC mill, Arduino kits.

Module 12: Automotive User Interface, Laser Engraving, and Arduino Integration

Objective: Design user interfaces for automotive applications, integrate laser engraving for UI elements, and use Arduino for sensor-based interactions.

Exercise:

1. Lecture (10 minutes): Fundamentals of automotive UI design, overview of laser engraving, and Arduino for sensor integration.
2. Design Session (30 minutes): Create a prototype for an automotive interface (e.g., dashboard layout).
3. **Laser Engraving (20 minutes):** Use laser engraving to create high-precision UI elements for the automotive interface.
4. **Arduino Integration (20 minutes):** Implement Arduino to simulate sensor-based interactions (e.g., temperature control, obstacle detection).
5. Testing (20 minutes): Test the prototype for usability and functionality.

6. Feedback Session (30 minutes): Present and discuss design improvements.

Materials and Tools: Automotive UI design tools, computers, laser engraving machine, Arduino kits.

IV. REFERENCE BOOKS:

1. Chee Kai Chua, Kah Fai Leong, "3D Printing and Additive Manufacturing: Principles and Applications".
2. Simon Knight, "Arduino for Beginners: Step-by-Step Guide to Arduino (Arduino Hardware & Software)".
3. Peter Smid, "CNC Programming Handbook".
4. Steven Wolfe, "Laser Cutting and Engraving".
5. Charles Platt, "Make: Electronics: Learning Through Discovery".

COURSE CONTENT

AIRCRAFT STRUCTURES								
IV Semester: AE								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AAED06	Core	L	T	P	C	CIA	SEE	Total
		3	0	-	3	40	60	100
Contact Classes: 48	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: 48			
Prerequisite: Mechanics of Solids								

I. COURSE OVERVIEW:

This course is designed to study the behavior of aircraft structural elements subjected to inertial, aerodynamic, and maneuver loads. Thin-walled beams, thin plates analysis are being conducted by energy methods. Structural idealization and load analysis on wing, fuselage, and landing gears are integral part of this course. The ultimate goal is to design and development of the new generation aircraft components. This course is a prerequisite for Analysis of Aircraft structures and Finite Element Methods/Analysis.

II. COURSE OBJECTIVES:

The students will try to learn:

- The application of mathematical principles to aircraft components to determine deflections and stresses under various loading conditions.
- The bending of thin-walled structures and the concept of structural idealization to transform complex structures to simple structures
- The concept of structural idealization and transformation of complex structures to simple structures.
- The behavior of wing, fuselage and landing gears under various loading conditions.

III. COURSE OUTCOMES:

After successful completion of the course, students will be able to:

- CO 1 Utilize the Impact Strength and Fatigue Strength concept for interpreting stresses due to axial, bending and torsional loads effect of inertia, Goodman and Soderberg relationship, and stresses due to combined loading, cumulative fatigue damage.
- CO 2 Choose Strain Energy and Columns concept for predicting the to axial, bending and Torsional loads, various end conditions, Euler's Column curve, Rankine's formula, and Column with initial curvature.
- CO 3 Inspect Bending of thin-walled beams for finding the Mechanical Behaviors.
- CO 4 Develop torsion and shear of thin plate for predicting the mechanical properties.
- CO 5 Illustrate the concepts General aspects of Shear stress distribution for interpreting end of a closed section beam, Thin-walled rectangular section beam subjected to torsion.
- CO 6 Make use of concept of Torsion of an arbitrary section beam, Distributed torque loading for determining the I-section beam subjected to torsion and Moment couple conditions.

IV. COURSE CONTENT:

MODULE-I: INTRODUCTION TO AIRCRAFT STRUCTURAL COMPONENTS AND ENERGY METHODS (10)

Aircraft Structural components and loads, functions of structural components, airframe loads; Types of structural joints, type of loads on structural joints; Aircraft inertia loads; Symmetric manoeuvre loads, gust loads. Monocoque and semi monocoque structures, stress in thinshells; Introductions to energy principles, castiglianos theorems, maxiwells reciprocal theorem, unit load method, Rayleigh Ritz method, total potential energy method, flexibility method.

MODULE –II: THIN PLATE THEORY, STRUCTURAL INSTABILITY (10)

Analysis of thin rectangular plates subject to bending, twisting, distributed transverse load, combined bending and in-plane loading: Thin plates having small initial curvature, energy methods of analysis. Buckling of thin plates:

Elastic, inelastic, experimental determination of critical load for a flat plate, local instability, instability of stiffened panels, failure stresses in plates and stiffened panels. Tension field beams- complete diagonal tension, incomplete diagonal tension, post buckling behavior.

MODULE –III: BENDING, SHEAR AND TORSION OF THIN-WALLED BEAMS (09)

Unsymmetrical bending: Resolution of bending moments, direct stress distribution, position of neutral axis; Deflections due to bending: Approximations for thin-walled sections, temperature effects; Shear loaded thin walled beams: General stress, strain and displacement relationships, direct stress and shear flow system, shear centre, twist and warping.

Torsion of beams of closed section: Displacements associated with Bredt-Batho shear flow; Torsion of open section beams; Warping of cross section, conditions for zero warping; Bending, shear, torsion of combined open and closed section beams.

MODULE –IV: STRUCTURAL IDEALIZATION (10)

Structural idealization: Principal assumptions, idealization of panel, effect on the analysis of thin-walled beams under bending, shear, torsion loading application to determining deflection of open and closed section beams. Fuselage frames - bending, shear and torsion

MODULE –V: ANALYSIS OF FUSELAGE, WING AND LANDING GEAR (09)

Wing spar and box beams, tapered wing spar, open and closed sections beams, beams having variable stringer areas; wings – three boom shell in bending, torsion and shear, tapered wings, deflections, cutouts in wings; Cutouts in fuselages; Fuselage frame and wing rib; principle of stiffener, web constructions. Landing gear and types; Analysis of landing gear

V. TEXT BOOKS:

1. T. H. G. Megson, “Aircraft Structures”, Butterworth-Heinemann Ltd, 5th Edition, 2012.
2. E. H. Bruhn, “Analysis and Design of Flight vehicles Structures”, Tri-state off set company, USA, 4th edition, 1965.

VI. REFERENCE BOOKS:

1. B. K. Donaldson, “Analysis of Aircraft Structures - An Introduction”, McGraw Hill, 3rd edition, 1993.
2. S. Timoshenko, “Strength of Materials”, Volumes I and II, Princeton D. Von Nostrand Co., Reprint, 1977.

VII. ELECTRONICS RESOURCES:

1. <https://as.wiley.com/WileyCDA/WileyTitle/productCd-1118806778.html>
2. <https://www.scribd.com/document/63588270/Aerospace-Propulsion-Systems>

VIII. MATERIALS ONLINE

1. Course template
2. Tutorial question bank
3. Tech talk topics
4. Open end experiments
5. Definitions and terminology
6. Assignments
7. Model question paper – I
8. Model question paper - II
9. Lecture notes
10. E-learning readiness videos (ELRV)
11. Power point presentation

COURSE CONTENT

AIRCRAFT PROPULSION AND TURBO MACHINERY								
IV Semester: AE								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AAED07	Core	L	T	P	C	CIA	SEE	Total
		3	0	-	3	40	60	100
Contact Classes: 48	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: 48			
Prerequisite: Thermodynamics and Heat Transfer								

I. COURSE OVERVIEW:

An aircraft propulsion system is a machine that produces thrust to push an aircraft forward. This course introduces various aircraft propulsion systems, and their performance analysis. The course discusses the operating principles of the aircraft engine's major components such as inlets, compressors, turbines, and nozzles. The design parameters, performance characteristics, and the factors influencing them are also addressed.

II. COURSE OBJECTIVES:

The students will try to learn:

- The fundamentals of air-breathing propulsion system, their operating principles, and function of an individual component.
- The geometry of low inlets, combustion chambers, and factors affecting their performance.
- The establishment of flow through various inlets and nozzles under different operating conditions.
- The operating principles of various compressors, turbines and performance characteristics under different flight conditions.

III. COURSE OUTCOMES:

After successful completion of the course, students will be able to:

- CO 1 Compare the operating principles of various gas turbine engines and their components for selecting the suitable engine as per the mission requirements.
- CO 2 Utilize the thrust equation and engine cycle analysis for achieving the required performance.
- CO 3 Apply the knowledge of flow through various inlets, and nozzles under various operating conditions for selecting the suitable inlets and nozzle as per the mission requirement.
- CO 4 Compare the different types of combustion chambers for identifying the design variables affecting their performance.
- CO 5 Summarize the operating principles of various compressors, turbines for their suitable selection.
- CO 6 Make use of the performance characteristics and efficiencies of different compressors and turbines for identifying a suitable combination.

IV. COURSE CONTENT:

MODULE-I: AEROSPACE ENGINES CLASSIFICATION AND THERMODYNAMIC CYCLE ANALYSIS (10)

Description and function of gas generator, Classifications of Aerospace Engines-Turbo jet Engines, Turboprop, Turboshaft, Turbo fan Engines, Propfan Engines Advanced Ducted Fan, Classification of Jet Engines-Ramjet, Pulsejet, Scramjet, Turboramjet, Turbo rocket -Thrust Equation, Factors Affecting Thrust Jet Nozzle Air Speed Mass Air Flow Altitude, Ram Effect, Ideal cycle analysis- turbo jet, turbofan, real cycle analysis- turbojet, Engine Performance Parameters Propulsive

Efficiency Thermal Efficiency Propeller Efficiency Overall Efficiency Take off Thrust Specific Fuel Consumption, Aircraft Range, Endurance Factor Specific Impulse.

MODULE –II: INLETS AND COMBUSTION CHAMBERS (10)

Internal flow and stall in subsonic inlets, relation between minimum area ratio and external deceleration ratio, diffuser performance, supersonic inlets, operating conditions of supersonic inlet, starting problem on supersonic inlets, shock swallowing by area variation; Classification of combustion chambers, Supersonic Combustion Chamber Combustion Process Chemistry of Combustion, important combustion parameters. Pressure losses; combustion efficiency; combustion intensity. Factors affecting combustion chamber design, and operation, flame stabilization, Cooling, Material, Aircraft Fuels, Emissions and Pollutants.

MODULE –III: NOZZLES (09)

Governing Equations, Theory of flow in isentropic nozzles, nozzles and choking, nozzle throat conditions, nozzle efficiency, losses in nozzles.

Over expanded and under expanded nozzles, Nozzle design considerations: fixed and variable geometry nozzles, after burning nozzles, thrust vectoring, thrust reversal Classification of Thrust Reverser Systems.

MODULE –IV: COMPRESSORS (10)

Principle of operation of centrifugal compressor and axial flow compressor, work done and pressure rise, types of velocity triangles, degree of reaction, free vortex and constant reaction designs of axial flow compressor, Basic design parameters , Centrifugal Stress, Tip Mach number, fluid deflection, design process for compressor, Blade design, Cascade measurements-Choosing the Type of Airfoil, Stage performance Blade Efficiency and Stage Efficiency, performance characteristics of centrifugal and axial flow compressors-single stage, multistage compressor, Stall and Surge, Surge Control Methods Multi spool compressor ,variable vanes, air bleed.

MODULE –V: TURBINES (09)

Principle of operation of axial flow turbines, comparison of axial flow compressors and turbines, limitations of radial flow turbines, work done and pressure rise, velocity triangles, degree of reaction, free vortex and constant angle designs, preliminary design- main design step, aerodynamics design performance characteristics, losses and efficiency, profile loss, annulus loss secondary flow loss, tip clearance loss, turbine blade cooling.

V. TEXT BOOKS:

1. Hill, P.G. & Peterson, C.R. “Mechanics & Thermodynamics of Propulsion” Addison Wesley Longman INC, 1999.
2. Mattingly J.D., “Elements of Propulsion: Gas Turbines and Rocket”, AIAA, 1991.

VI. REFERENCE BOOKS:

1. Cohen, H. Rogers, G.F.C. and Saravanamuttoo, H.I.H. “Gas Turbine Theory”, Longman, 1989
2. Oates, G.C., “Aero thermodynamics of Aircraft Engine Components”, AIAA Education Series, New York, 1985.

VII. ELECTRONICS RESOURCES:

1. <https://as.wiley.com/WileyCDA/WileyTitle/productCd-1118806778.html>
2. <https://www.scribd.com/document/63588270/Aerospace-Propulsion-Systems>

VIII. MATERIALS ONLINE

1. Course template
2. Tutorial question bank
3. Tech talk topics
4. Open end experiments
5. Definitions and terminology
6. Assignments

7. Model question paper – I
8. Model question paper - II
9. Lecture notes
10. E-learning readiness videos (ELRV)
11. Power point presentation

COURSE CONTENT

AERODYNAMICS								
IV Semester: AE								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AAED08	Core	L	T	P	C	CIA	SEE	Total
		3	0	-	3	40	60	100
Contact Classes: 48	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: 48			
Prerequisite: Fluid Dynamics								

I. COURSE OVERVIEW:

Aerodynamics deals with the study of the flow of air about a body, and the body can be an airplane, missiles, helicopters, and any other vehicles. This course will delve into several key areas, including the principles airflows and generation of aerodynamic forces. The concept of this course is relevant to a wide variety of applications ranging from sailboats, automobiles, birds, insects, and other aircrafts. This course will enable learners to gain a fundamental understanding of concepts and models used to aerodynamically analyze and some classical theories which are useful for design of aircraft components. It forms an essential corner stone for mechanical, chemical, and aerospace engineers and plays a pivotal role in the study of airflows not only for engineers but also for medical domain.

II. COURSE OBJECTIVES:

The students will try to learn:

- The fundamental knowledge on basics of aerodynamics and aerodynamic characteristics of wings, airfoils.
- The mathematical model for lift and drag coefficient of finite wing and wing of infinite aspect ratio.
- The flow over non-lifting bodies from method of singularities and investigate the interference effect
- The effect of viscosity and boundary layer growth over various shaped geometry and its control.

III. COURSE OUTCOMES:

After successful completion of the course, students will be able to:

- CO 1 Illustrate the aerodynamic flows with velocity potential for non-lifting and lifting flow based on fundamental laws of aerodynamics using Kutta-Joukowski theorem.
- CO 2 Make use of thin airfoil theory to ascertain aerodynamic characteristics, centre of pressure, and generation of lift force for the infinite aspect ratio wing using Kutta trailing edge condition.
- CO 3 Develop the vortex model for flow past of finite wing to ascertain drag using Kelvin and Helmholtz theorem, and Biot-Savart's law.
- CO 4 Demonstrate the influence of taper, twist, sweep back, and Delta wings for the generation of the aerodynamic forces using Source Panel, Vortex panel and Vortex lattice methods.
- CO 5 Illustrate the effect of interference due to wing body, propeller and other non-lifting surfaces for the total aerodynamic characteristic's estimation using Cauchy-Reiman relations and Kutta-Joukowski transformation.
- CO 6 Compare the basic concepts of Viscous flow model with Inviscid flow model for the finite wing by using Displacement body model and Wall transpiration model.

IV. COURSE CONTENT:

MODULE-I: INTRODUCTORY TOPICS FOR AERODYNAMICS (10)

Potential flow, velocity potential, stream function, Laplace equation, flow singularities-Uniform flow, source, sink, doublet, Vortex, Non lifting and lifting flow over a cylinder Kutta-Joukowski theorem.

MODULE –II: THIN AEROFOIL THEORY (10)

Aerofoil nomenclature, aerodynamic characteristics, Centre of pressure and aerodynamic Centre; Wing of infinite aspect ratio, $CL-\alpha$ - diagram for a wing of infinite aspect ratio, generation of lift, starting Vortex, Kutta's trailing edge condition; Thin aerofoil theory; High lift airfoils, High lift devices. Use of XFLR5 for simulation of lift on different aerofoils and compare.

MODULE –III: FINITE WING THEORY (12)

Vortex motions, vortex line, vortex tube, vortex sheet; Circulation; Kelvin and Helmholtz theorem; Biot-Savart's law, applications, Rankine's vortex; Flow past finite wings, vortex model of the wing and bound vortices; Induced drag; Prandtl's lifting line theory; Elliptic wing.

Influence of taper and twist applied to wings, effect of sweep back wings; Delta wings, primary and secondary vortex; Elements of lifting surface theory. Source Panel Vortex panel and Vortex lattice methods. Use of XFLR5 for simulation of effects of taper, sweep, and delta wing for demonstration.

MODULE –IV: FLOW PAST NON-LIFTING BODIES AND CONFORMAL TRANSFORMATION (08)

Flow past non lifting bodies, method of singularities; Wing-body interference; Effect of propeller on wings and bodies and tail unit; Flow over airplane as a whole. Potential, Cauchy-Reiman relations, Complex conformal transformation, Kutta-Joukowski transformation.

MODULE –V: VISCOUS EFFECT IN AERODYNAMIC FLOWS (08)

Inviscid flow model, Displacement effect: normal mass flow matching, normal mass flux in real flow; Improved Inviscid Flow Model: Displacement body model, Wall transpiration model, Wake modeling, Improved flow model advantages, Viscous decambering stall mechanism; Consideration in flow model selection.

V. TEXT BOOKS:

1. E. L. Houghton and P. W. Carpenter, "Aerodynamics for Engineering Students", Edward Arnold Publishers Ltd., London, 5th edition, 1982,
2. J. D. Anderson, "Fundamentals of Aerodynamics", McGraw Hill Book Co., New York, 5th edition, 1985.
3. Mark Drela "Flight Vehicle Aerodynamics" The MIT Press, Cambridge, Massachusetts, London, England, 2014.

VI. REFERENCE BOOKS:

1. L. J. Clancy, "Aerodynamics", Pitman, 1st edition, 1986.
2. L. H. Milne, S. Thomson, "Theoretical Aerodynamics", Dover, 2nd edition, 1985
3. K. Karamcheti, "Principles of Ideal-Fluid Aerodynamics", Krieger Pub Co; 2nd edition, 1980.

VII. ELECTRONICS RESOURCES:

1. <https://www.loc.gov/rr/scitech/tracer-bullets/aerodynamicstb.html>
2. <https://www.myopencourses.com/subject/aerodynamics-2>
3. <https://tocs.ulb.tu-darmstadt.de/211658790.pdf>
4. <https://www.princeton.edu/~stengel/MAE331Lecture3.pdf>

VIII. MATERIALS ONLINE

1. Course template
2. Tutorial question bank

3. Tech talk topics
4. Open end experiments
5. Definitions and terminology
6. Assignments
7. Model question paper – I
8. Model question paper - II
9. Lecture notes
- 10.E-learning readiness videos (ELRV)
- 11.Power point presentation

COURSE CONTENT

FLIGHT MECHANICS								
IV Semester: AE								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AAED09	Core	L	T	P	C	CIA	SEE	Total
		3	0	-	3	40	60	100
Contact Classes: 48	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: 48			
Prerequisite: Fluid Dynamics								

I. COURSE OVERVIEW:

Flight mechanics is the science that investigates the performance of the aircraft as applied to flight vehicles and to provide a clear understanding of related topics, specifically on aerodynamics, propulsion, performance, stability and flight controls. The course introduces the fundamental principles of aerodynamics and propulsion for aircraft performance in classical flying stages. This course is the point of confluence of other disciplines with aeronautical engineering and the gateway to aircraft design.

II. COURSE OBJECTIVES:

The students will try to learn:

- I. The fundamental principles of aerodynamics and propulsion for aircraft performance in classical flying stages.
- II. The different regimes of aircraft and performance requirements at various atmospheric conditions.
- III. The mathematical models for various types of maneuvers, safety requirements during takeoff, landing for better performance and stability.
- IV. Longitudinal and directional parameters with the help of the linearized equations of aircraft motion.

III. COURSE OUTCOMES:

After successful completion of the course, students will be able to:

- CO 1 Demonstrate the mission profiles of simple cruise, commercial transport and military aircrafts for getting the airplane performance characteristics.
- CO 2 Explain the cruise performance of an airplane in relation with range and endurance with different types of aircraft engines.
- CO 3 Develop the aircraft man oeuvre performance to perform in turn, pull-up and pull-down man oeuvres by considering limitations of power for military and civil aircrafts
- CO 4 Compare the various landing distances such as discontinued landing, bulk landing for better stability and control of the aircraft.
- CO 5 Examine the different types of dynamic modes in longitudinal, lateral and directional motion for the aircraft and their influence on dynamic stability and safety.
- CO 6 Apply the advance theories of flight dynamics in design of modern control airplane control systems for enhancing aircraft performance, Modern control systems and autopilot system

IV. COURSE CONTENT:

MODULE-I: INTRODUCTION TO AIRCRAFT PERFORMANCE (10)

The role and design mission of an aircraft; Performance requirements and mission profile; Aircraft design performance, the standard atmosphere; Off-standard and design atmosphere; Measurement of air data; Air data computers; Equations of motion for performance - the aircraft force system; Total airplane drag- estimation, drag reduction methods; The propulsive forces, the thrust production engines, power producing engines, variation of thrust, propulsive power and specific fuel consumption

with altitude and flight speed; The minimum drag speed, minimum power speed; Aerodynamic relationships for a parabolic drag polar.

MODULE –II: CRUISE PERFORMANCE (09)

Maximum and minimum speeds in level flight. Range and endurance with thrust production, and power producing engines. Cruise techniques - constant angle of attack, constant Mach number; constant altitude, methods- comparison of performance. The effect of alternative fuel flow laws, weight, altitude and temperature on cruise performance. Cruise performance with mixed power-plants

MODULE –III: CLIMB, DESCENT and MANOEUVRE PERFORMANCE (11)

Climb and descent techniques, safety considerations, performance analysis- maximum climb gradient, climb rate. Energy height and specific excess power, optimal climbs - minimum time, minimum fuel climbs. Measurement of climb performance. Descent performance in aircraft operations. Effect of wind on climb and descent performance.

Accelerated motion of aircraft - equations of motion- the manoeuvre envelope. Longitudinal manoeuvres- the pull-up, push over manoeuvres. Lateral manoeuvres- turn performance- turn rates, turn radius, limiting factors. Manoeuvre boundaries, Manoeuvre performance of military aircraft, transport aircraft.

MODULE –IV: TAKEOFF and LANDING (09)

Estimation of take-off distances. The effect on the take-off distance wrt weight, wind, runway conditions, ground effect. Take off safety factors, The estimation of landing distances, the discontinued landing, baulked landing air safety procedures and requirements on performance The effect on the landing distance, of weight, wind, runway conditions, ground effect. Fuel planning, fuel requirement, trip fuel, reserve and tankering.

MODULE –V: FLIGHT PERFORMANCE CONSIDERATIONS IN TACTICAL MISSILE DESIGN (09)

Flight performance envelope, Equations of motion modeling, driving parameters for flight performance, Cruise flight performance, Steady state flight, Flight trajectory shaping, Turn radius, Coast flight performance, Boost flight performance, Intercept lead angle and velocity, Comparison with performance requirements.

V. TEXT BOOKS:

1. Turner, M.J.L., “Rocket and Spacecraft Propulsion”, MIT Press, 2nd Edition, 2022.
2. Anderson, J.D. Jr., “Aircraft Performance and Design”, International edition McGraw Hill, 1st Edition, 1999, ISBN: 0-07-001971-1. 2. Eshelby, M.E., “Aircraft Performance theory and Practice”, AIAA Education Series, AIAA, 2nd Edition, 2000, ISBN: 1-56347-398-4.
3. Nelson, R.C, “Flight Stability and Automatic Control”, Tata McGraw Hill, 2nd Edition, 2007, ISBN 0-07-066110- 3.

VI. REFERENCE BOOKS:

1. McCormick, B.W, “Aerodynamics, Aeronautics and Flight Mechanics”, John Wiley, 2nd Edition, 1995, ISBN: 0-471- 57506-2.
2. Yechout, T.R. et al., “Introduction to Aircraft Flight Mechanics”, AIAA Education Series, AIAA, 1st Edition, 2003, ISBN: 1-56347-577-4.
3. Shevel, R.S., “Fundamentals of Flight”, Pearson Education, 2nd Edition, 1989, ISBN: 81-297-0514-
4. Schmidt, L.V., “Introduction to Aircraft Flight Dynamics”, AIAA Education Series, 1st Edition, 1998, ISBN A56347-226-0.

VII. ELECTRONIC RESOURCES:

1. www.myopencourses.com/subject/flight-dynamics-i-airplane-performance.
2. www.scribd.com/doc/185026212/Introduction-to-Flight-Third-Edition-by-John-D-Anderson.Jr

3. www.scribd.com/book/282507871/Performance-and-Stability-of-Aircraft
4. www.scribd.com/doc/203462287/Aircraft-Performance-NPTEL
5. www.nptel.ac.in/courses/101106041

VIII. MATERIALS ONLINE

1. Course template
2. Tutorial question bank
3. Tech talk topics
4. Open end experiments
5. Definitions and terminology
6. Assignments
7. Model question paper – I
8. Model question paper - II
9. Lecture notes
10. E-learning readiness videos (ELRV)
11. Power point presentation



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

AEROSPACE MATERIALS AND PRODUCTION TECHNOLOGY								
IV Semester: AE								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AAED10	Core	L	T	P	C	CIA	SEE	Total
		3	0	-	3	40	60	100
Contact Classes: 48	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: 48			
Prerequisite: Mechanics of Solids								

I. COURSE OVERVIEW:

The subject aircraft Production Technology encompasses on providing good theoretical background and a sound practical knowledge of Aircraft materials and Manufacturing process used for producing aircraft components to the engineering students. It plays a vital role for producing aircraft components with minimum cost and with longer service. The subject deals with combination of materials and heat treatment process, fabrication of composite materials, manufacturing process such as casting, welding, sheet metal forming, riveting process, machining process, automation and jigs and fixtures which are widely employed in industries.

II. COURSES OBJECTIVES:

The students will try to learn

- Applications of Aircraft materials and their alloys in aerospace industry.
- The manufacturing process employed for different materials and quality inspection techniques.
- The working principles and applications of different machining processes and their advantages and disadvantages.
- The importance of composites and their applications for aerospace applications

III. COURSE OUTCOMES:

After successful completion of the course, students should be able to:

- CO1 Identify the properties of flight vehicle materials and their significant for producing parts of airplanes.
- CO2 Use of principles and techniques of casting and metal joining for determining their suitability.
- CO3 Identify the types of heat treatment process and quality inspections methods for improving life of aircraft parts.
- CO4 Demonstrate the riveting and forming process in making fuselage and wings of aircrafts.
- CO5 Make use of suitable machining process to achieve the specified product performance and design criterion by considering cost and time
- CO6 Identify the suitable fabrication process for composite materials to have high strength to its weight

IV. COURSE CONTENT:

MODULE – I: AIRCRAFT MATERIALS (10)

Classification of aircraft materials, properties of materials and their significance, Factors affecting the selection of material for different parts of airplanes,

Metals And Alloys applications; Aluminum Titanium and Steel alloys, Effect of alloying elements, Nickel and cobalt based alloys, refractory materials, ceramics, and super alloys.

Heat Treatment of Materials: annealing, normalizing, hardening and tempering of Aluminum and steel, Corrosion - Types of Corrosions - Prevention - Protective Treatments

MODULE –II: AIRCRAFT COMPOSITES (09)

Classification, characteristics of composite materials, volume fraction, laminated composites, particulate composites and fibrous composites. Types of reinforcements, their shape and size, production and properties of fiber and glass reinforced plastics. Application of Composite materials in aerospace industries.

MODULE – III: CASTING AND WELDING AND INSPECTION TECHNIQUES (09)

General principles of various casting processes Sand casting, die-casting, centrifugal casting, investment casting, Shell molding types; Principles and equipment used in arc welding, gas welding, resistance welding, solid, laser welding, and electron beam welding, soldering and brazing techniques. NDT, testing.

MODULE –IV: CONVENTIONAL AND UNCONVENTIONAL MACHINING PROCESSES (10)

General working principles, applications and operations of lathe, shaper, milling machines, grinding, drilling machine, computer numeric control machining;

Working principles and applications of abrasive jet machining, ultrasonic machining, Electric discharge machining and electro chemical machining, laser beam, electron beam, plasma arc machining.

MODULE –IV ADDITIVE MANUFACTURING (10)

Introduction to Additive Manufacturing (AM)- Evolution, Steps and Classification of AM processes, advantages and disadvantages, Types of materials for AM, Stereo lithography (SL), AM Manufacturing processes- Fused Deposition Modelling (FDM), Selective laser Sintering (SLS) and Direct Metal Deposition (DMD). Postprocessing- Support Material Removal, Surface Texture, Accuracy and Aesthetic Improvement.

V. TEXT BOOKS:

1. S. Kalpakjian, Steven R. Schmid, “Manufacturing Engineering and Technology”, Addison Wesley , 5th Edition, 1991.
2. G. F. Titterton, Aircraft Materials and Processes, 5/e, Sterling Book House, 1998.
3. S. C. Keshu, K. K Ganapathy, “Aircraft production technology and management”, Interline Publishing House, Bangalore, 3rd edition, 1993.
4. C. P. Paul, A. N. Jinoop, “Additive Manufacturing”, 1st edition-2021, McGraw Hill India

VI. REFERENCE BOOKS:

1. S. C. Keshu, K. K Ganapathy, “Air craft production techniques”, Interline Publishing House, Bangalore, 3rd edition, 1993.
2. R. K. Jain, “Production technology”, McGraw Hill, 1st edition, 2002.
3. O. P. Khanna, M. Lal, “Production technology”, Dhanpat Rai Publications, 5th edition, 1997.
4. L. Gupta, Advanced Composite Materials, 2/e, Himalayan Books, 2006.

VII. ELECTRONICS RESOURCES:

1. <https://nptel.ac.in/courses/112107145/>
2. <https://nptel.ac.in/courses/112105126/>
3. https://books.google.co.in/books?id=6wFuW6wufTMC&redir_esc
4. <https://royalmechanicalbuzz.blogspot.in/2015/04/manufacturing-engineering-by-kalpakjian.html>

VIII. MATERIALS ONLINE:

1. Course template
2. Tutorial question bank
3. Tech talk topics

4. Open end experiments
5. Definitions and terminology
6. Assignments
7. Model question paper – I
8. Model question paper - II
9. Lecture notes
10. E-learning readiness videos (ELRV)
11. Power point presentation

COURSE CONTENT

AEROSPACE STRUCTURES LABORATORY								
IV Semester: AE								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AAED11	Core	L	T	P	C	CIA	SEE	Total
		-	-	2	1	40	60	100
Contact Classes: Nil	Tutorial Classes: Nil	Practical Classes: 45			Total Classes: 45			
Prerequisite: Aircraft Structures								

I. COURSE OVERVIEW:

Analytical tools are introduced in the course from an engineering standpoint. The course emphasizes the significance of aircraft structures while attempting to provide a foundational understanding of analytical techniques. By encouraging undergraduate students to work on projects related to the structural analysis of thin-walled structural components, wings, fuselage, and landing gears, aircraft structural laboratories are utilized to improve student learning.

II. COURSE OBJECTIVES:

The students will try to learn:

1. The basic knowledge on the mechanical behaviour of materials like aluminium, mild steel, and cast iron.
2. The crack detection using various NDT methods and also discuss the changing strength due to these defects.
3. The concept of locating the shear centre for open and closed sections of beams.
4. Buckling strength of both long and short columns using different elastic supports

III. COURSE OUTCOMES:

After successful completion of the course, students should be able to:

- CO1 Use the amount of load for stress and strain to illustrate the characteristics of materials subjected to tensile loads in engineering applications.
- CO2 Examine the purpose of aerospace structural design, to illustrate the deflections of beams subjected to transverse loads under various end circumstances.
- CO3 Utilize the Beam test rig and apply Maxwell's reciprocal theorem to simplify the analysis through symmetry.
- CO4 Inspect the crucial buckling loads of columns that are subjected to compression loads so that constructions can be designed efficiently under a range of end circumstances.
- CO5 Determine which columns in the South-well plot are subject to axial loads in order to determine the critical loads.
- CO6 Make use of a beam's asymmetrical bending behavior in terms of aerospace structure design.

EXERCISES FOR AEROSPACE STRUCTURES LABORATORY

1. Getting Started with Exercises

Introduction

The goal of this course is to familiarise students with current tools and methods while giving them the chance to conduct two subject experiments pertaining to solid and structural mechanics. The intention is to provide an illustration of how typical materials behave and how structural combinations used in aerospace applications work.

Data Recording and reports

Students must record their experimental values in the provided tables in this laboratory manual and reproduce them in the lab reports. Reports are integral to recording the methodology and results of an experiment. In engineering practice, the laboratory notebook serves as an invaluable reference to the technique used in the lab and is essential when trying to duplicate a result or write a report. Therefore, it is important to learn to keep accurate data. Make plots of data and sketches when these are appropriate in the recording and analysis of observations. Note that the data collected will be an accurate and permanent record of the data obtained during the experiment and the analysis of the results.

1.1 Landing Gears

1.1.1 Investigation of Landing Gears

- Landing gear is located under the belly of the plane consisting of a wheel and strut to soften impact with the ground and may be retractable into the fuselage. Tricycle type wheels are common for general aviation with one wheel at the front and two behind or the reverse for tailwheels with two wheels at the front of the plane and one under the tail as shown in Fig. 1.1.1.
- All parts of an airplane are crucial for conducting safe flight. Pilots' huge responsibility is ensuring all aircraft components are in excellent condition before embarking on their flight journey.
- Actual landing of an aircraft is shown in Fig. 1.1.2.

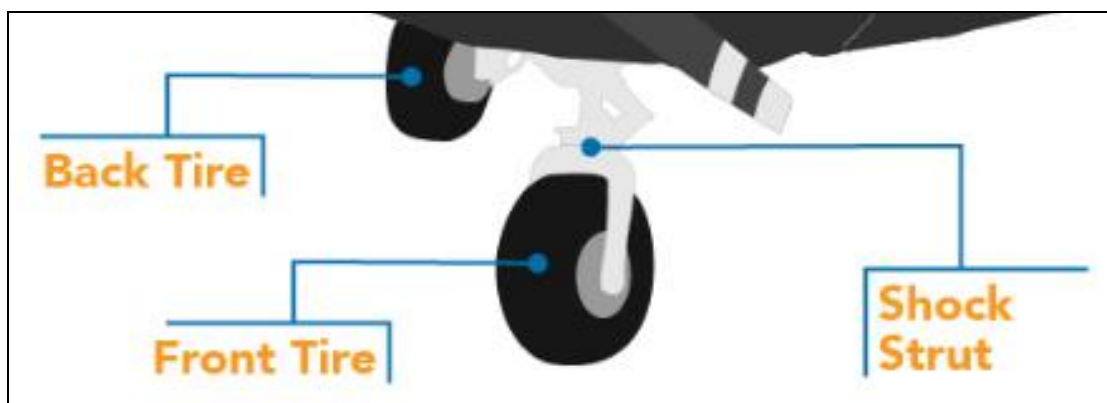


Fig. 1.1.1: Landing gear

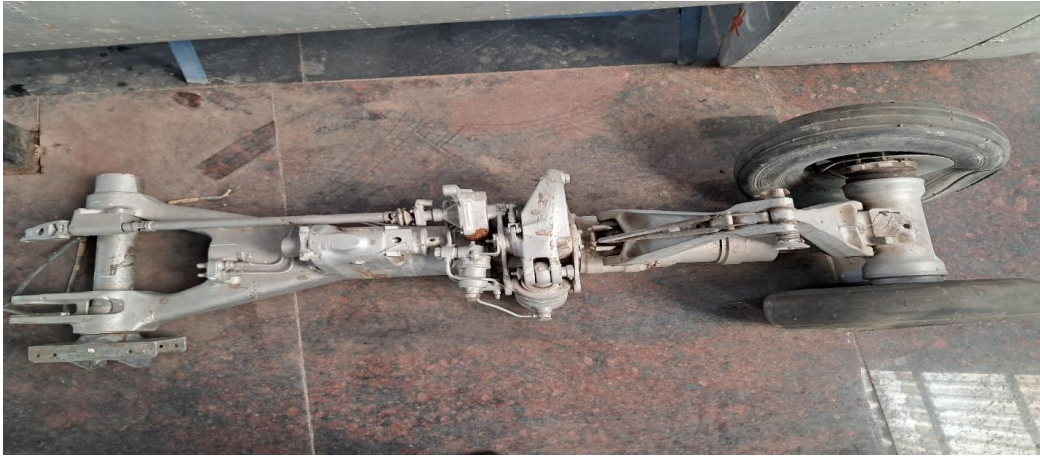


Fig. 1.1.2: Landing gear of an actual aircraft

Try

1. Scrutinize the landing gear of mechanism and assess their structural integrity.
2. Examine the landing gear of fighter jet and conducting a detailed inspection of assembly of landing gear.

1.2 VERIFICATION OF MAXWELL'S RECIPROCAL THEOREM

1.2.1 Maxwell's Reciprocal Theorem

- I. Determine the deflection of a simply supported beam by keeping load at point 1 and finding the deflection at point 2 as shown in Fig. 1.2.1.
- II. Determine the deflection by keeping load at point 2 and find the deflection at point.
- III. Compare the deflection for both the cases.



Fig. 1.2.1: Beam apparatus

Try

1. Find the deflection at point C ($L/4$) due to unit load at point D ($3L/4$).
2. Find the deflection at point D ($3L/4$) due to unit load at point C ($L/4$).
3. Compare the deflection values for at both C and D.
4. Perform the experiments across a range of diverse loads, systematically analyze and assess the effects of varying conditions.

2. BUCKLING TEST

2.1 Determine the critical buckling loads by compression tests on long columns

- I. Determine the critical buckling load for a long column with the help of buckling of struts equipment as shown in Fig. 2.1.
- II. Determine the deflection by applying the various loads on a long column.



Fig. 2.1: Loading and buckling of struts equipment

Try

1. Determine the critical buckling of strut at the open lever condition.
2. Find the critical buckling of strut at closed lever condition.
3. Compare the open and closed lever condition and analyze the results.

3. COMPRESSION TEST

3.1 Determine the critical buckling loads, Southwell plot by compression tests on short columns

- I. Determine the critical buckling load for a short column with the help of buckling of struts equipment as shown in Fig. 3.1.
- II. Determine the deflection by applying the various loads on a short column.
- III. Draw the Southwell plot, load versus deflection.



Fig. 3.1: Loading and buckling of struts equipment for compression test

Try

1. Explore the critical buckling of strut at the open lever condition.
2. Find the critical buckling of strut at closed lever condition.
3. Compare the open and closed lever condition and analyze the results.

4. BENDING TEST

4.1 Determine the unsymmetrical bending of a beam

- I. Determine the unsymmetrical bending of a given open section beam by using the apparatus shown in Fig. 4.1
- II. Determine the unsymmetrical bending of a given closed section beam by using the apparatus shown in Fig. 4.1
- III. Find the unsymmetrical bending of an open section beam by changing the eccentricity.
- IV. Find the unsymmetrical bending of a closed section beam by changing the eccentricity.

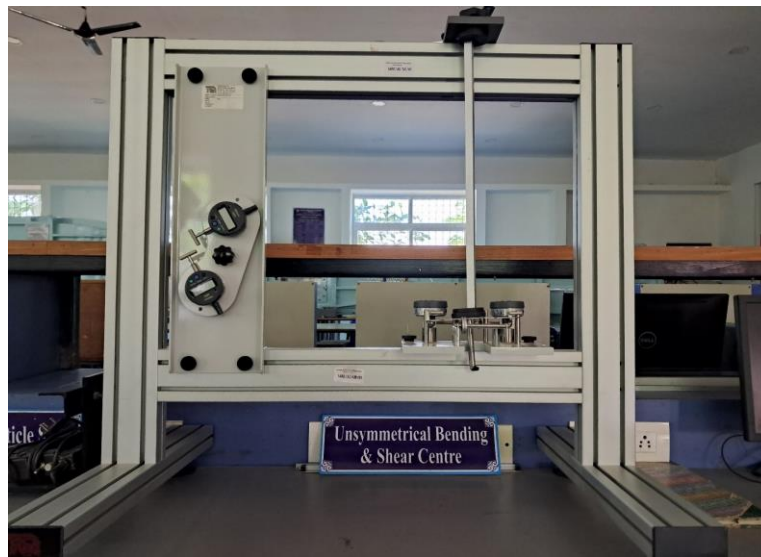


Fig. 4.1: Apparatus required for the unsymmetrical bending test

Try

1. Conduct a thorough analysis on the asymmetric bending characteristics exhibited by an open section (L Shape).
2. Determine the unsymmetrical bending of channel section (C Shape).
3. Perform the unsymmetrical bending characteristics exhibited by rectangular section.

5. SHEAR CENTRE FOR OPEN SECTION

5.1 Determination of shear centre of an open section beam

- I. Determine the shear centre of a given open section beam by using the apparatus shown in Fig. 5.1.
- II. Find the shear centre of an open section beam by changing the eccentricity.



Fig. 5.1: Equipment for determination of shear centre of an open section beam

Try

1. Find the shear centre of an open section beam L shape configuration.
2. Determine the shear centre of an open section beam I-shape configuration.
3. Compare the shear centre values of a L and I shape configurations and examine the results.

6. SHEAR CENTRE FOR CLOSED SECTION

6.1 Determination of shear centre of a closed section

- I. Determine the shear centre of a given closed section beam by using the apparatus shown in Fig. 6.1.
- II. Find the shear centre of a closed section beam by changing the eccentricity.

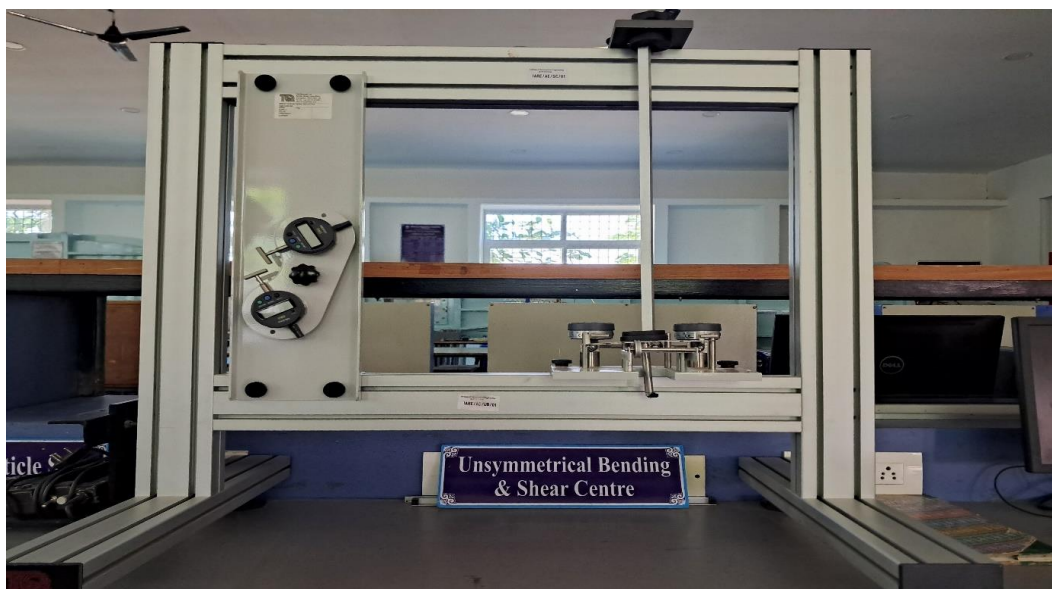


Fig. 6.1: Equipment for determination of shear centre of a closed section beam

Try

1. Find the shear centre of closed section beam hollow pipe configuration.
2. Determine the shear centre of closed section beam solid circular pipe configuration.
3. Determine the shear centre of closed section beam of square shape.
4. Investigate the shear centre of wood circular rod and compare the results with others.

7. SHEAR STRESS OF RIVETED JOINTS

7.1 Determine the shear strength of riveted joint (double riveted Zig-Zag lap joint) between two given metals

- I. Determine the shear strength of a riveted joint as shown in Fig. 7.1 using the Universal Testing Machine (UTM) machine as given in Fig. 7.2.
- II. Consider d_2 as diameter of the rivet, apply the shear stress S_s and measure the strength of the rivet.
- III. The rivet shearing: The rivet might shear as the figure illustrates in Fig 7.1. The joint's maximal force tolerance to stop this failure is
$$P^2 = S_s * ((\pi/4) * d_2^2) \text{ for lap joint}$$
$$P^2 = 2S_s * ((\pi/4) * d_2^2) \text{ for single strap butt joint}$$
Where S_s =allowable shear stress of the rivet material

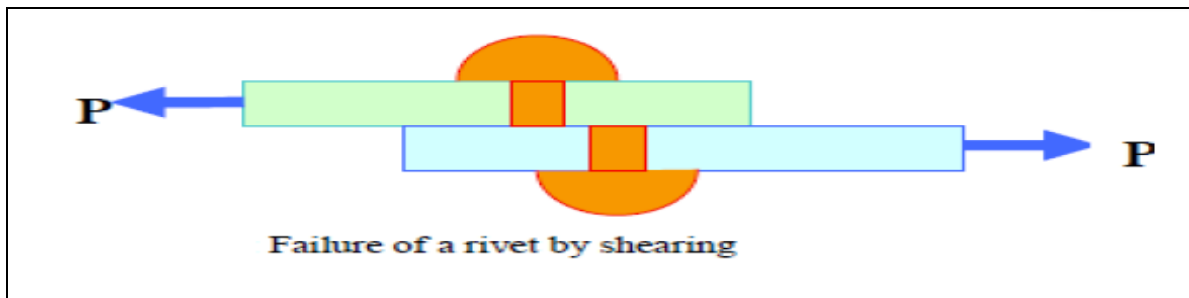


Fig. 7.1: Riveted joint



Fig. 7.2: UTM machine for finding shear stress

Try

1. Determine the shear strength of the riveted triple joint between two given metals.
2. Explore the shear strength of the Lap joint.
3. Assess the shear strength of a riveted double joint connecting two dissimilar-sized metals.

8. SANDWICH PANEL TENSION TEST (Composite Materials)

8.1 Fabricate and determine the young's modulus of Sandwich Panel (Composite Materials)

- I. Fabricate the Sandwich panel by using composite material.
- II. Determine the young's modulus of the Sandwich panel by using the UTM machine as shown in Fig. 8.1.



Fig. 8.1: UTM Machine for finding young's modulus

Try

1. Fabricate the sandwich panel with inner material of wooden fiber.
2. Find the strain in the laminated sandwich panel.
3. Determine the Young's Modulus of the laminated sandwich panel.
4. Fabricate the sandwich panel with different materials and find the strain and Young's Modulus of the composite.

9. NON-DESTRUCTIVE TESTING (Dye-Penetration Method)

9.1 Dye penetration test

- I. Find the crack propagated in the given specimen by using the Dye Penetrant Test and Setup shown in Fig. 9.1 & 9.2.
- II. Inspect the surface crack in the given specimen by using the Dye Penetrant Test.

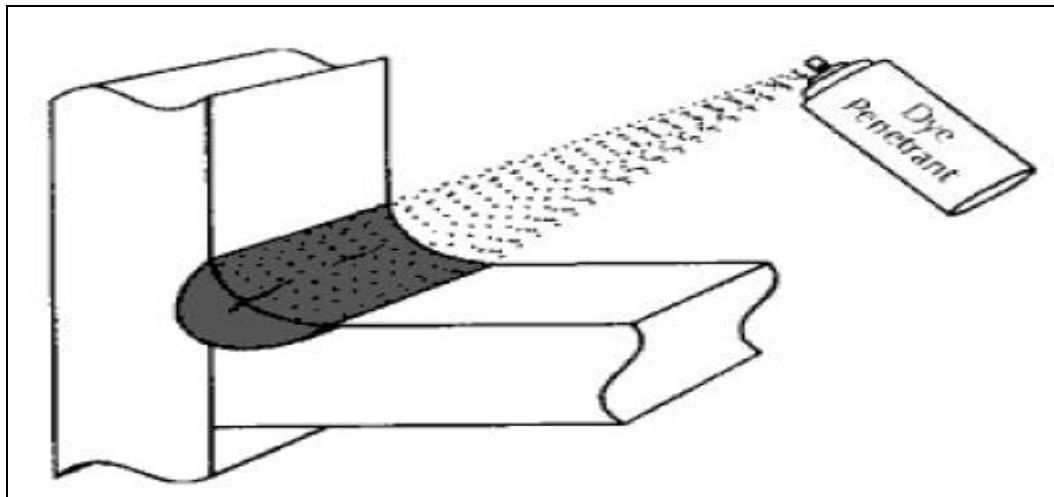


Fig. 9.1: Dye Penetrant Test



Fig. 9.2: Dye Penetrant Test Setup

Try

1. Examine the components of a fighter jet aircraft to identify the onset of initial cracks.
2. Inspect the cracks with non-destructive test for a landing gear.
3. Investigate the propagating cracks in an airframe.
4. Investigate the solid V and T shaped specimen for finding the cracks

10. NON-DESTRUCTIVE TESTING (Magnetic particle inspection)

10.1 Detect the flaws in the given specimen by conducting non-destructive testing procedures using magnetic particle inspection

- I. Inspect the given specimen and find if any cracks are present by using Magnetic Particle Inspection (MPI) Test shown in Fig. 10.1.
- II. Identify the growth of the cracks by using Magnetic Particle Inspection (MPI) Test setup shown in Fig. 10.2.

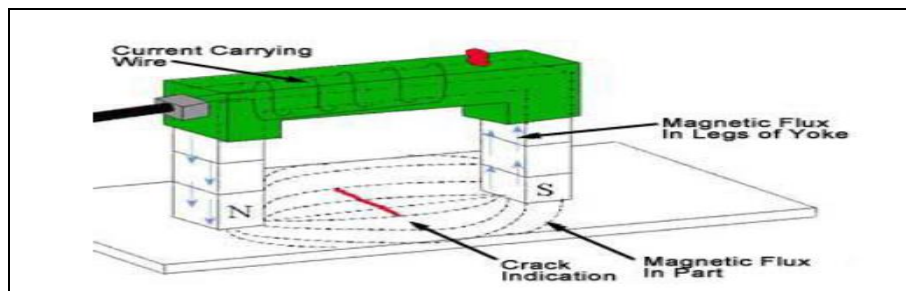


Fig. 10.1: MPI Test



Fig. 10.2: MPI Test Setup

Try

1. Scrutinize the components of a fighter jet aircraft to identify the onset of initial cracks.
2. Check the cracks with non-destructive test for a landing gear.
3. Investigate the propagating cracks in an airframe.
4. Explore the solid V and T shaped specimen for finding the cracks.

11. NON-DESTRUCTIVE TESTING (Ultrasonic techniques)

11.1 Non-destructive test using ultrasonic test

- I. Take the given specimen and match with the C.R.T screen.
- II. Find out the position of flaw using transducer as shown in Fig. 11.1.



Fig.11.1: Ultrasonic Test

Try

1. Observe the components of a fighter jet aircraft to identify the onset of initial cracks.
2. Examine the cracks with non-destructive test for a landing gear.
3. Inspect the propagating cracks in an airframe.
4. Explore the solid V and T shaped specimen for finding the cracks.

12. VIBRATION TEST

12.1 Determine of natural frequency of beams under free and forced vibration

- I. Determine the natural frequency of beam under free vibration as shown in Fig.12.1.
- II. Find the natural frequency of a beam by applying the force by the hammer as shown in Fig.12.2.
- III. By using the trigger as shown in Fig.12.3, find the natural frequency of a beam.



Fig. 12.1: Showing Initial Set-up



Fig 12.2: Showing impact loading with hammer

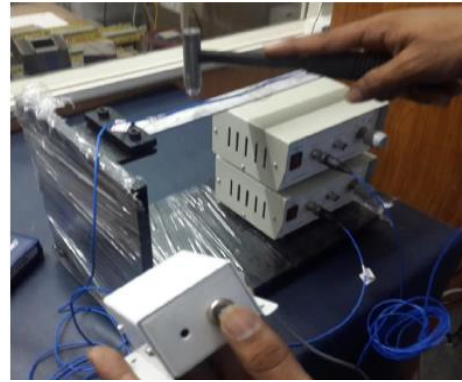


Fig 12.3: Showing impact loading with trigger

Try

1. Determine the natural frequency 10, 100, 1000Hz under free vibration with hammer.
2. Find the natural frequency 10, 100, 1000Hz under free vibration with trigger.
3. Examine the natural frequency 10, 100, 1000Hz under forced vibration with trigger.
4. Assess the natural frequency 10, 100, 1000Hz under forced vibration with hammer.

13 Aircraft Wings

13.1 Study about the Aircraft Wings

- I. A wing is a kind of fin that moves through fluids such as air by creating lift.
- II. As a result, wings have streamlined cross sections that function as airfoils when subjected to aerodynamic forces.
- III. A wing's lift-to-drag ratio indicates how aerodynamically efficient it is.
- IV. One to two orders of magnitude more lift can be produced by a wing at a given speed and angle of attack than total drag.
- V. A high lift-to-drag ratio means that a much smaller effort is needed to get the wings airborne at a sufficient lift.
- VI. Scrutinize the ribs and assessing their structural integrity, aerodynamic features, and overall condition to ensure optimal performance and safety as shown in Fig.13.1.



Fig. 13.1: Aircraft wing's cut out

Try

1. Examine the spars and conducting a detailed inspection for their structural integrity.
2. Investigate the assembly of the webs and stringers in an aircraft wing.

14. AIRFRAMES

14.1 Investigation of aircrafts parts like fuselages, cockpit, engine, propeller

Fuselages: The plane's body, or fuselage, holds the aircraft together, with pilots sitting at the front of the fuselage, passengers and cargo in the back.

Cockpit: The cockpit is the area at the front of the fuselage from which a pilot operates the plane. The cockpit contains the:

- Instrument panel: This is similar to a car's dashboard, providing the pilot with information about the flight, the engine and the circumstances of the aircraft. Depending on the aviation electronics (avionics) installed in an aircraft this may be on an interactive screen or using the typical '6 Pack' for key pieces of information.
- Flight controls: In the cockpit are two seats, one for the pilot and the other for the co-pilot.
- Pilot seats: In the cockpit are two seats, one for the pilot and the other for the co-pilot.
- Rudder pedals: Rudder pedals control yaw in flight and are used for steering on the ground during a taxi.
- Overhead panel: The overhead panel contains aircraft systems, such as air conditioning, electrical, fuel and hydraulics.
- Side consoles: Side consoles are for communication instruments and documentation, depending on the aircraft.

Engine: The engine(s), or powerplant, of an aircraft creates thrust needed for the plane to fly.

Propeller: An aircraft's propeller(s) are airfoils, similar to a wing, installed vertically to create thrust to drive the plane forward. Attached to the engine, they spin quickly, creating lift from the pressure difference they create, only instead of this lift causing the plane to move upwards, it drives the plane forward creating thrust. This thrust and forward motion in turn causes air to pass over the wings, creating the vertical lift.

Airframe containing aircraft parts like fuselages, cockpit, engine, propeller is shown in Fig. 14.1.



Fig. 14.1: Airframe

Try

1. Visualize the assembly of the propeller and wing.
2. Scrutinize the empennage of the aircraft.
3. Study the fuselage and wing interaction and assembly.
4. Examine the engine configuration of the Rollce-Royce.

IV. TEXTBOOKS:

1. R.K Bansal, Strength of Materials., Laxmi publications, 5th Edition, 2012.
2. T. H. G. Megson, Aircraft Structures for Engineering Students., Butterworth-Heinemann Ltd,5th Edition, 2012.
3. Gere, Timoshenko, Mechanics of Materials., McGraw Hill, 3rd Edition,1993.

V. REFERENCE BOOKS:

1. Peery,D.J. and Azar,J.J., Aircraft Structures, 2nd edn, McGra-Hill, 1982, ISBN0-07-049196-8
2. Bruhn.E.H, Analysis and Design of Flight Vehicles Structures, Tri-state Off-set Company, USA, 1965
3. Lakshmi Narasaiah, G., Aircraft Structures, BS Publications, 2010.

VI. ELECTRONICS RESOURCES:

1. https://akanksha.iare.ac.in/index?route=course/details&course_id=1645.

VII. MATERIALS ONLINE

1. Course Template
2. Laboratory Manual



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

AERODYNAMICS AND PROPULSION LABORATORY								
IV Semester: AE								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AAED12	Core	L	T	P	C	CIA	SEE	Total
		-	-	2	1	40	60	100
Contact Classes: Nil	Tutorial Classes: Nil	Practical Classes: 45			Total Classes: 45			
Prerequisite: Aerodynamics and Propulsion								

I. COURSE OVERVIEW:

The aerodynamics and propulsion laboratory course typically involves hands-on experiments and practical applications to reinforce the theoretical concepts learned in aerodynamics and propulsion courses. The lab is designed to provide students with a deeper understanding of the principles related to the behaviour of aerodynamic concepts and propulsion systems used in aircrafts. This course offers a wide range of applications in aerodynamics and propulsion such as measurement of lift, drag, moment, boundary layer and thrust measurements. It forms an essential cornerstone for aerospace engineers, plays a pivotal role in the efficient design and testing of various aircraft components.

II. COURSE OBJECTIVES:

The students will try to learn:

- I. The wind tunnel calibration and associated instrumentation.
- II. The measurement of lift and drag coefficients of various aerodynamic components
- III. The measurement of performance characteristics of compressor, blower, propeller and nozzle
The thrust measurement and performance calculation of gas turbine engine.

III. COURSE OUTCOMES:

After successful completion of the course, students should be able to:

- CO1 Make use of various calibration techniques for assessing the flow quality in wind tunnel test section.
- CO2 Examine the pressure distribution over airfoil, cylinder and flat plate for predicting their aerodynamics characteristics.
- CO3 Utilize the six-component force balancer for deducing the forces and moments of aircraft model and hence obtaining the aircraft performance and stability.
- CO4 Determine the pressure, temperature across each component gas turbine engine for predicting its thrust and performance characteristics.
- CO5 Examine the performance characteristics of compressor, blower, propeller and nozzle for their efficient design
- CO6 Identify the flash point, fire point and calorific value of different fuels for their suitability in aerospace applications.

IV. COURSE CONTENT

EXERCISES FOR AERODYNAMICS AND PROPULSION LABORATORY

Note: Students are encouraged to bring their own laptops for laboratory practice sessions to plot the graph and do the calculation using excel.

1. Getting Started Exercises

Introduction to Low-Speed Wind Tunnel

All the experiments in the aerodynamics part will be carried out in suction type- open circuit low speed wind tunnel (Fig.1).

Specification

Test section size = 600 X 600 x 2000 mm

Maximum speed in test section= 54 m/s

Fan speed range= 60-1350 rpm



Fig 1: Low speed open circuit wind tunnel

Instrumentation

- Multitube manometer
- Inclined manometer
- Pitot tube
- Pitot-Static tube
- Wake rake
- Boundary layer rake
- Six component balance
- Digital anemometer

Introduction to Axial Flow Gas Turbine

The CM14 axial flow turbine engine, has been integrated into a sturdy metal frame that holds it firmly, while enabling accurate measurement of the thrust produced by the engine. Electronic preprogrammed controller constantly supervises the engine, ensuring safe operating conditions at all times. The engine is controlled via the software, which provides users with a graphical interface for real-time monitoring and operation (Fig.2).

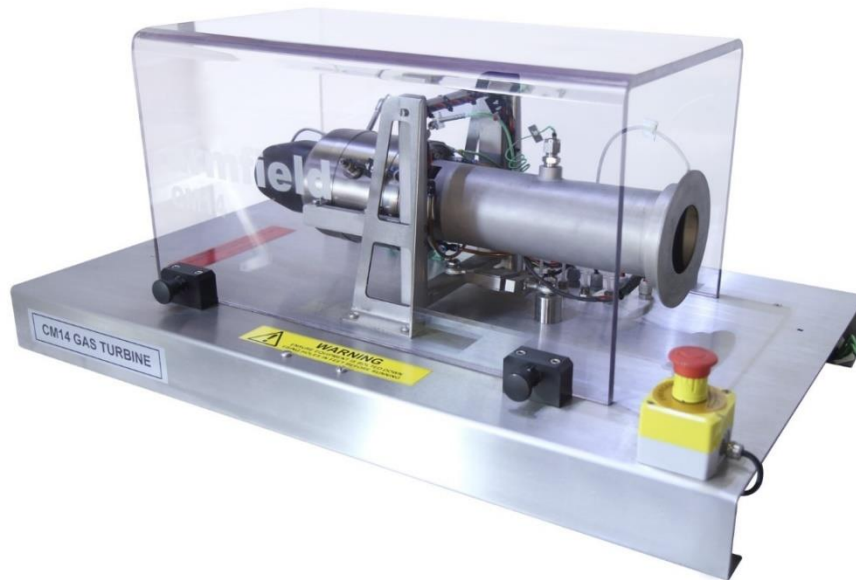


Fig. 2: CM14 axial flow turbine engine

Specification

- Typical fuel: One of the following choices
 - Paraffin
 - Jet A-1
 - JP-4/Kerosene
- Exhaust gas temperature: 800°C typical
- Mass flow: 450 g/s
- Ignition system: Glow plug
- Compressor type: Single-stage radial
- Turbine type: Single-stage, low-mass axial flow
- Engine rpm: 105,000rpm typical
- Engine mount: Single pivot point

1.1 Measurement of pressure and flow velocity

Measure the wind tunnel static, dynamic and total pressure with different flow velocity.

Try

1. Measure the flow velocity
2. Measure the static pressure
3. Measure the total pressure

2. Exercises on calibration and pressure distribution over cylinder

2.1 Calibration of subsonic wind tunnel

Determine the true speed and flow angularity in the wind tunnel test section for propeller rpms 100 to 1300 in steps of 100 and plot RPM vs velocity, and RPM vs flow angularity (Fig. 3).



Fig. 3: Wind tunnel calibration

Try

1. Change the propeller rpm to 200 to 500 in steps of 50 and plot RPM vs velocity.
 2. Change the propeller rpm to 600 to 900 in steps of 100 and plot RPM vs angularity.
 3. Change the propeller rpm to 1000 to 1400 in steps of 200 and plot RPM vs velocity
-

2.2 Pressure distribution over cylinder

Measure the pressure distribution over a circular cylinder having span of 600 mm, diameter of 50 mm, and Reynolds number of 3×10^5 (laminar) and determine coefficient of pressure and drag coefficient. Also compare the experimental and theoretical c_p values (Fig. 4).

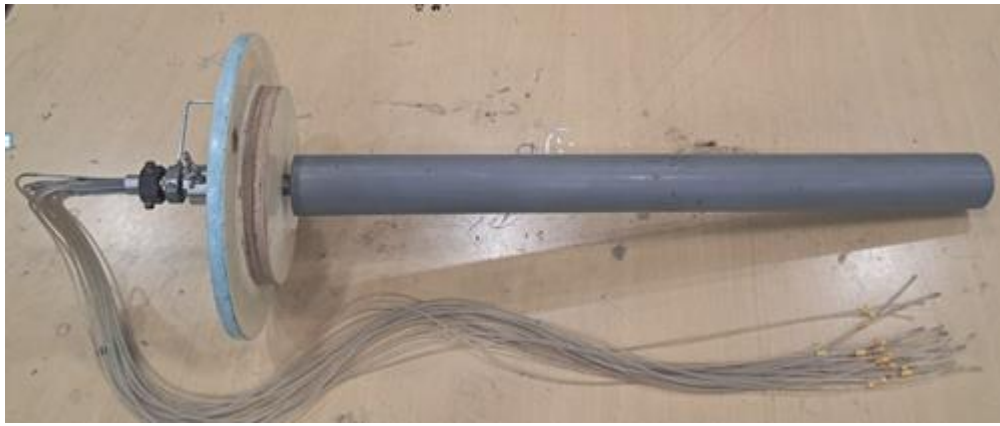


Fig.4: Cylinder model

Try

1. Change the Reynolds number to 5.5×10^5 (transition) and determine the C_p distribution, drag coefficient
2. Change the Reynolds number to 7×10^5 (turbulent) and determine the C_p distribution, drag coefficient

3. Exercises on pressure distribution and flow visualization

3.1 Pressure distribution over symmetrical airfoil

Measure the pressure distribution over a NACA 63(2)-215 symmetrical airfoil having span of 600 mm, chord length of 150 mm, Reynolds number of 3×10^5 (laminar) and determine coefficient of pressure, lift, and drag coefficient at different angle of attack. Also plot α vs C_L , α vs C_D , and x/c vs C_p , (Fig. 5).

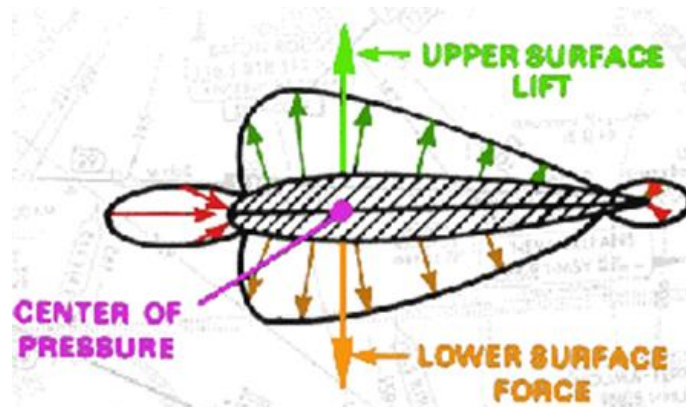


Fig.5: Symmetrical airfoil pressure distribution

Try

1. Determine the c_l , c_d , c_p for NACA 63(2)-215 cambered airfoil at Reynolds number of 3×10^5
2. Change the Reynolds number to 7×10^5 (turbulent) and determine the c_l , c_d , c_p .
3. Determine c_l , c_d , and c_p for the NACA Symmetrical airfoil (Fig. 5).

3.2 Flow visualization (Circular cylinder)

Determine the wake width and flow separation point of a circular cylinder having span of 600 mm, diameter of 50 mm, at different Reynolds number (Fig. 6).



Fig.6: Smoke flow visualization behind the circular cylinder [Sharad, 2021]

Try

3. Repeat the same experiment for a NACA 63(2)-215 symmetrical airfoil having different Reynolds number and angle of attack.
4. Repeat the same experiment for a NACA 0015 symmetrical airfoil having different velocity and angle of attack.

3.3 Flow visualization over a car model

Tuft/ smoke flow visualization on a car model with different speed (Fig. 7, 8).



Fig .7: Smoke flow visualization over the model car



Fig. 8: Tuft flow visualization over the model car

Try

1. Visualization of flow pattern at the rear end of the Fastback and Square back car model.
2. Flow visualization with rear spoilers on a car.
3. Influence of the slant angle in car back (Fig.9)

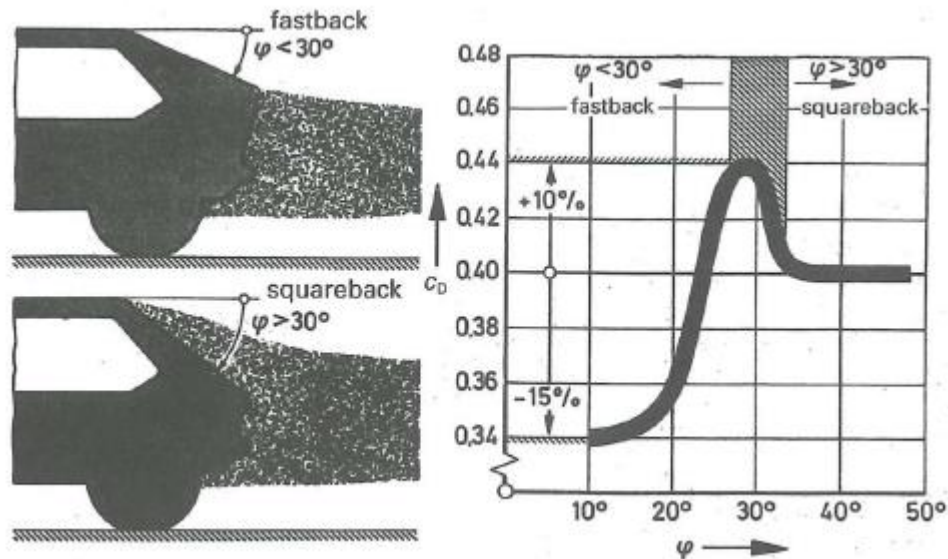


Fig.9: Effect of slant angle and flow pattern on a car back[Hucho, 1998]

4. Exercises on wake analysis

4.1 Wake analysis of a circular cylinder / Car model

Determine the drag coefficient of a circular cylinder having span of 600 mm and diameter of 50mm at Reynolds number of 1.0×10^6 by the wake survey method (Fig.10).

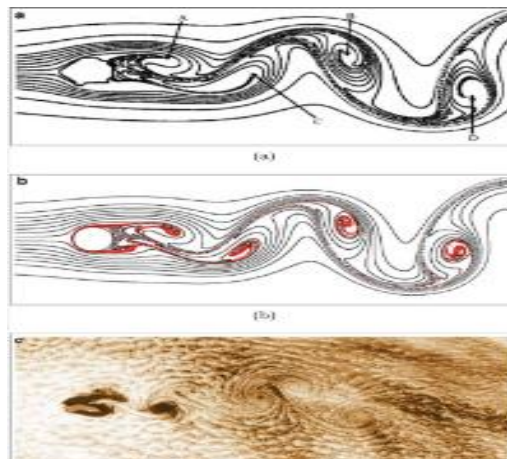


Fig .10: Wake analysis of circular cylinder

Try

1. Change the Reynolds number to 3.0×10^4 (laminar) and determine drag coefficient
2. Change the Reynolds number to 1.5×10^6 (turbulent) and determine drag coefficient
3. Wake analysis if the car model at given Reynolds number

4.2 Wake analysis of a symmetrical airfoil

Determine the drag coefficient of a NACA 63(2)-215 symmetrical airfoil having span of 600 mm and chord length of 150mm, Reynolds number of 3×10^5 (laminar) and angle of attack by the wake survey method (Fig.11).

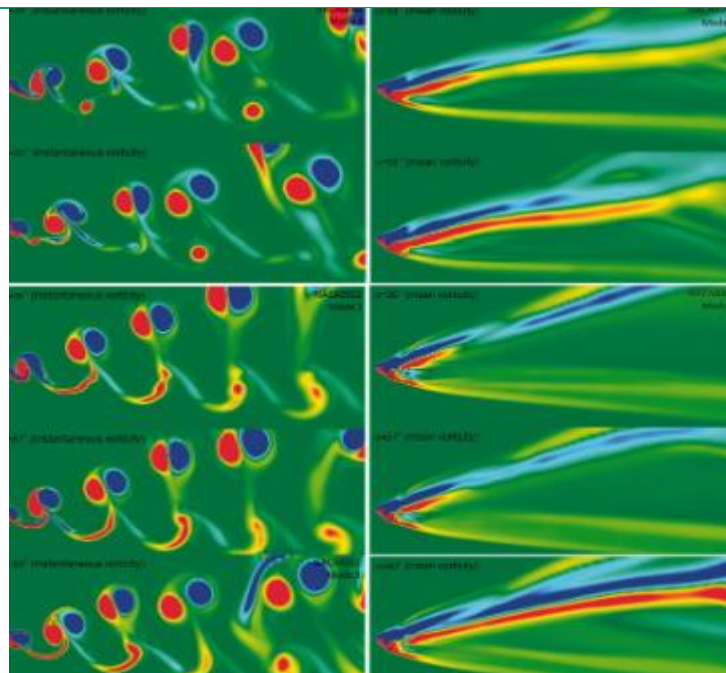


Fig.11: Wake pattern of symmetric airfoil

Try

1. Repeat the same experiment for a NACA 63(2)-215 cambered airfoil having angle of attack.
2. Change the Reynolds number to 7×10^5 (turbulent) and determine drag coefficient.
3. Flow field on a backward facing step of a car model (Fig.12).

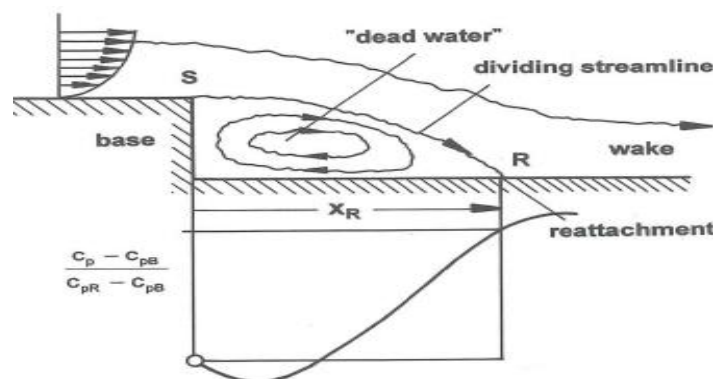


Fig.12: Flow field on a backward facing step of a car model [Hucho, 1998]

5. Exercises on force measurement

5.1 Force measurement of Aircraft model

Determine the lift, drag, side force, pitching moment, rolling moment, yawing moment of an aircraft model having Reynolds numbers of 3×10^5 , 5.5×10^5 and 7×10^5 at a given angle of attack using six component balance (Fig.13).



Fig.13: Six component balance with display and strain gauges

Try

1. Repeat the same experiment for different angle of attack and determine all the forces, moments
2. Introduce vortex generator on the wing surface and determine all the forces, moments
3. Determination of forces and moments over a car model to study its performance and stability.

5.2 Force measurement of Car model

Determine the lift, drag, side force, pitching moment, rolling moment, yawing moment of given car model having Reynolds numbers of 3×10^5 , and 7×10^5 at a zero-degree angle of attack using six component balance (Fig.14)

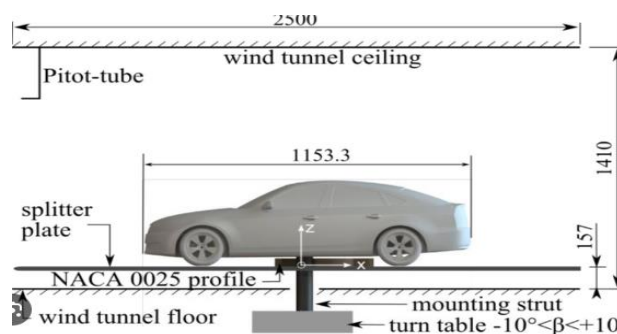


Fig.14: Experimental setup of scaled down car model

Try

1. Undertake the experiment for different orientations to determine the forces, and moments at $Re\ 5 \times 10^5$.
2. Determine the side force and yawing moment with variation of side slip angle (Fig. 15).
3. Determine the stability of a given car model with varying velocity.

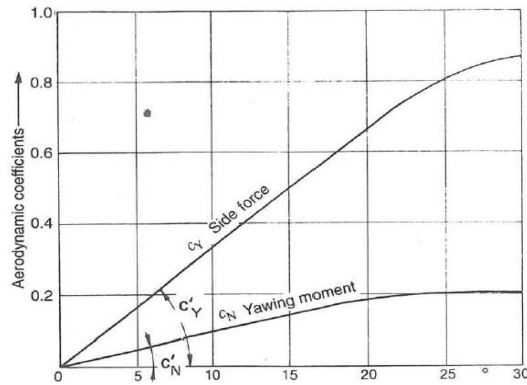


Fig.15: Plot of side force and yawing moment with different yawing angle [Hucho, 1998]

6. Exercises on flow over a flat plate

6.1 Boundary layer measurements on flat plate

Determine the boundary layer thickness at different station of a flat plate having characteristic length of 2000 mm at a given test section velocity. Also compare experimental results with theoretical results of laminar and turbulent approximation (Fig. 16).

Try

1. Introduce vortex generators at the leading edge of flat plate and determine the boundary layer thickness at different stations.
2. Determine the boundary layer thickness at different station of a symmetrical airfoil having characteristic length of 30 mm at a given test section velocity of 30 m/s.
3. Determine the boundary layer thickness at different station of a symmetrical airfoil having characteristic length of 50 mm at a given test section velocity of 20 m/s.



Fig.16: Flat plate

6.2 Prediction of skin friction drag by boundary layer measurement

Determine the skin friction of a flat plate by measuring the boundary layer. Also compare experimental results with computational results of laminar and turbulent approximation. (Fig. 16).

Try

1. Introduce vortex generators at the leading edge of flat plate and determine the shear stress.
2. Measure the velocity gradient in vertical plane and calculate the skin friction at different speed..
3. Determine the boundary layer thickness of a flat plate having characteristic length of 50 mm at a test section velocity of 30 m/s.

7. Exercises on gas turbine parameters calculation

7.1 Prediction of Gas Turbine parameters

Estimate the thrust output CM14 – axial flow gas turbine engine, using the momentum equation, and compare this result to the measured thrust (Fig.17).

Try

1. Calculate the thrust for different fuel flow rate.
2. Determine the thrust for different airflow rate.
3. Evaluate the power developed of the gas turbine at different rpm.

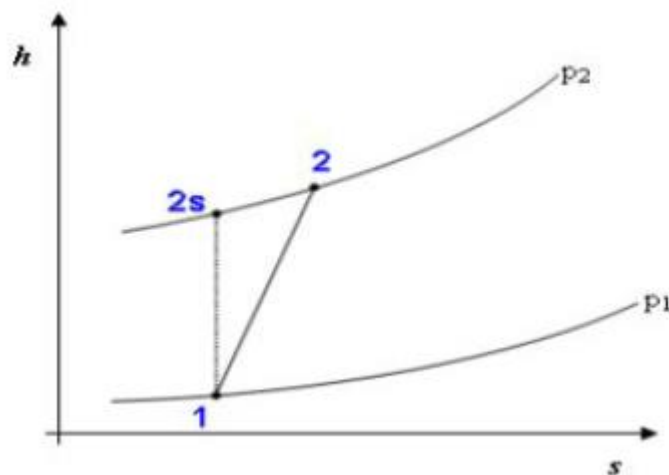


Fig.17:Enthalpy and entropy plots of Gas turbine

7.2 Prediction of Gas Turbine temperature variation

Measure the temperature variation of output CM14 – axial flow gas turbine engine, using thermal sensor, and compare this result to the measured computer value (Fig.17).

Try

1. Measure the temperature at inlet of the turbine.
2. Find the temperature at compressor inlet zone.
3. Evaluate the temperature at exhaust area.

8. Exercises on gas turbine performance diagrams and efficiency

8.1 Prediction of Gas Turbine performance

Estimate the actual thermodynamic cycle experienced by the flow throughout the CM14 – axial flow gas turbine engine and represent it in the form of the Temperature vs. Specific Entropy diagram. Also calculate thermal, propulsive and overall efficiency of engine (Fig. 18)

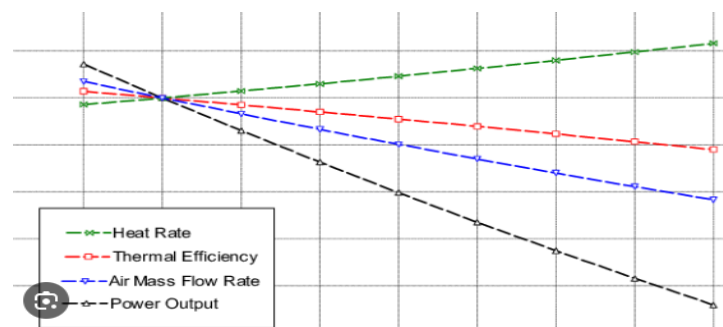


Fig.18: Performance characteristics of Gas turbine

Try

1. Plot the Temperature vs. Specific Entropy diagram for different fuel flow rate
2. Plot the Temperature vs. Specific Entropy diagram for different air flow rate
3. Plot the Temperature vs. Specific Entropy diagram for different air fuel mix flow rate

8.2 Prediction of Gas Turbine efficiency

Estimate the actual thermodynamic cycle efficiency by the flow throughout the CM14 – axial flow gas turbine engine and represent it in the form of the Temperature vs. Specific Entropy diagram (Fig. 18).

Try

1. Calculate thermal, propulsive and overall efficiency of engine.
2. Estimate thermal, propulsive and overall efficiency of engine.
3. Plot the Temperature vs. Specific Entropy diagram for different air fuel mix flow rate.

9. Exercises on gas turbine components efficiency

9.1 Prediction of Gas Turbine components efficiency

Estimate the efficiencies of intake, compressor, combustion chamber, turbine and nozzle of CM14-axial flow gas turbine engine for given fuel flow rate (Fig. 19)

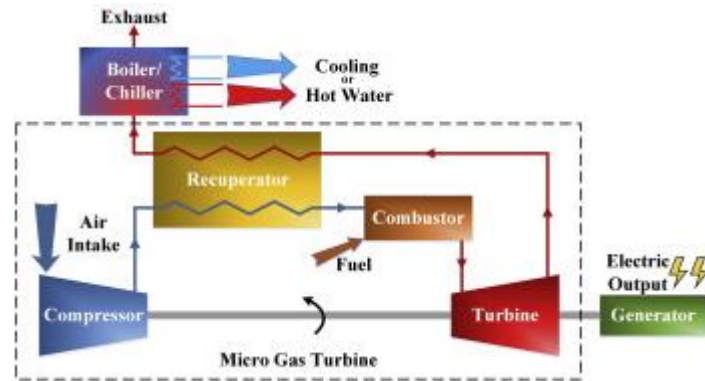


Fig.19: Micro gas turbine components

Try

1. Calculate the efficiency of gas turbine at different fuel flow rate.
2. Undertake turbine efficiency at different air flow rate.
3. Examine the turbine efficiency at different mixed flow rate.

9.2 Prediction of compressor efficiency

Estimate the efficiencies of compressor of CM14-axial flow gas turbine engine for given fuel flow rate (Fig. 19)

Try

1. Calculate the compressor or turbine efficiency at different fuel flow rate
2. Undertake the above analysis different air flow rate
3. Examine the above analysis different mixture flow rate

10. Exercises on blower test rig

10.1 Estimation of blower efficiency

Estimate the discharge, and suction head, of the centrifugal blower with forward vane impellor at different operating condition. Also plot Efficiency vs Discharge and Discharge vs Head (Fig. 20).



Fig.20: Blower test rig

Try

1. Undertake the above analysis for centrifugal blower with radial vane impellor.
2. Repeat the above analysis for centrifugal blower with backward vane impellor.
3. Repeat the above analysis for centrifugal blower with straight vane impellor.

10.2 Estimation of blower Performance

Estimate the delivery head and efficiency of the centrifugal blower with forward vane impellor at different operating condition. Also plot Efficiency vs Discharge and Discharge vs Head (Fig. 20).

Try

1. Carry out the above analysis for centrifugal blower with radial vane impellor.
2. Undertake the above analysis for centrifugal blower with backward vane impellor.

11. Exercises on compressor

11.1 Centrifugal compressor

Estimate the discharge, suction head, delivery head and efficiency of the centrifugal compressor with forward vane impellor at different operating condition. Also plot Efficiency vs Discharge and Discharge vs Head (Fig. 21).

Try

1. Repeat the above analysis for centrifugal compressor with half valve opening
2. Repeat the above analysis for centrifugal compressor with full valve opening
3. Repeat the above analysis for centrifugal compressor with 1/4 valve opening



Fig.21: Centrifugal compressor test rig

11.2 Axial flow compressor

Estimate the discharge, suction head, delivery head and efficiency of the centrifugal blower with forward vane impellor at different operating condition. Also plot Efficiency vs Discharge and Discharge vs Head.



Fig.22:Axial flow compressor test rig

Try

1. Repeat the above analysis for 1000 rpm (Fig. 22)
2. Repeat the above analysis for 1500 rpm (Fig. 22).
3. Repeat the above analysis for 2000 rpm(Fig. 22).

12.Exercises on nozzle Performance

12.1 Estimation of nozzle discharge and head

Estimate the discharge, and head across multiple stations. Also plot Discharge Vs Velocity.(Fig. 23).



Fig.23: Nozzle performance test rig

Try

1. Estimate the discharge, head, and efficiency for different inlet conditions.
2. Estimate the discharge, head, and thrust for different outlet conditions.
3. Estimate the efficiency, and thrust for ambient conditions.

12.2 Estimation of nozzle discharge and head

Estimate the, efficiency, thrust, wall pressure distribution and velocity variation across multiple stations. Also plot Discharge Vs Velocity.(Fig. 23).

Try

1. Estimate the efficiency and thrust for different inlet conditions.
2. Estimate the head, and thrust for different outlet conditions.
3. Estimate the efficiency, and thrust for ambient conditions.

13. Exercises on propeller test rig

13.1 Estimation of nozzle discharge and head

Estimate the thrust, and mass flow rate of air, of propeller at different RPMs. Also compare the theoretical propulsive efficiency and estimated propulsive efficiency.



Fig.24: Propeller test rig

Try

1. Repeat the above analysis by changing the pitch of propeller.
2. Undertake a case study on variable pitch propeller
3. Prepare a report for 5 best propeller aircrafts of the world.

13.2 Estimation of nozzle discharge and head

Estimate torque, and efficiency of propeller at different RPMs. Also compare the theoretical propulsive efficiency and estimated propulsive efficiency.

Try

1. Examine the discharge and head by changing the pitch of propeller.
2. Undertake a case study on variable pitch propeller.
3. Prepare a report for 5 best propeller aircrafts of the world with efficiency.

14. Exercises on Sports aerodynamics

14.1 Cricket ball aerodynamic analysis

Determine the effects of seam position, and roughness of cricket ball with different used overs.

Try

1. Undertake wind tunnel testing to find the effect of seam for fast bowler (Fig. 25).
2. Undertake wind tunnel testing to find the effect of seam for spinners (Fig. 26).
3. Undertake wind tunnel testing of used (old) cricket ball (Fig. 27).

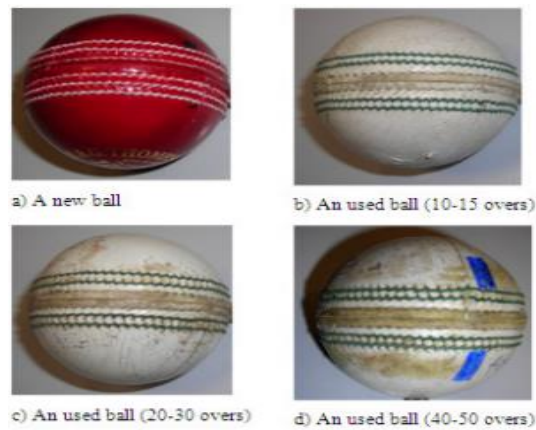


Fig.25: Types of ball used in cricket [Alam et. al, 2010]



Fig.26: Experimental setup to study cricket ball [Alam et. al, 2010]

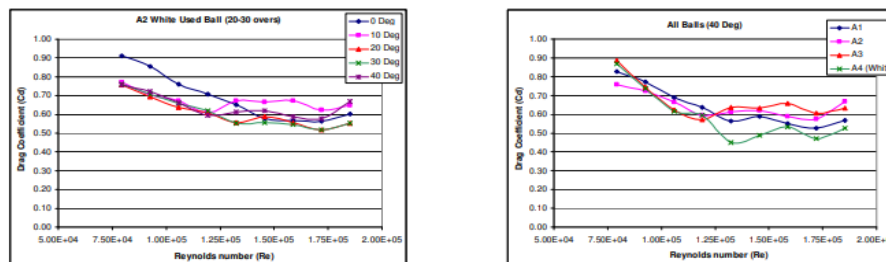


Fig.27: Results of to study cricket ball [Alam et. al, 2010]

14.2 Javelin throws aerodynamics

Determine the effects of wind direction on the range of javelin throw.

Try

1. Undertake the effect of tail wing on Javelin throw range (Fig. 28).
2. Undertake the effect of head wing on Javelin throw range (Fig. 29).
3. Undertake the effect of angle of throw on Javelin range (Fig. 28).

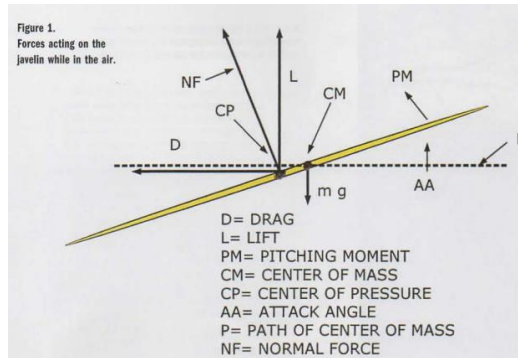


Fig.28: Forces acting on Javelin [A. Maheras, 2013]

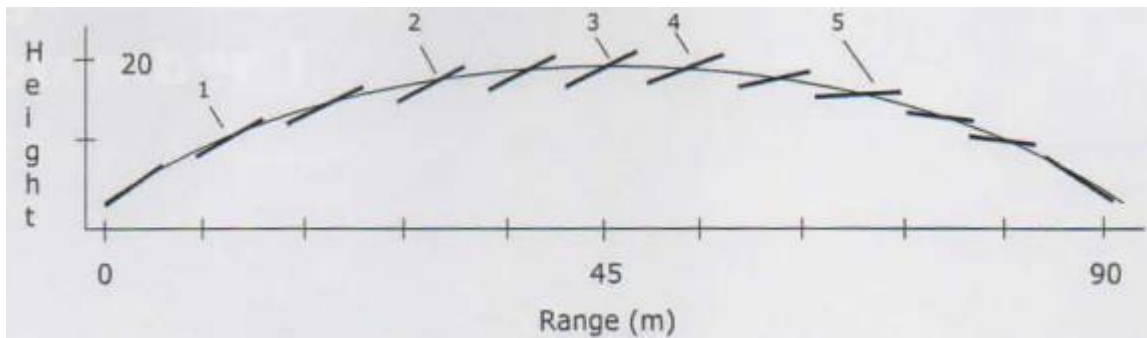


Fig.29: Javelin trajectory [A. Maheras, 2013]

14.3 Study of Badminton Shuttlecock aerodynamics

Determine the aerodynamic drag with variation of the Reynolds number for synthetic and natural feather shuttlecocks.

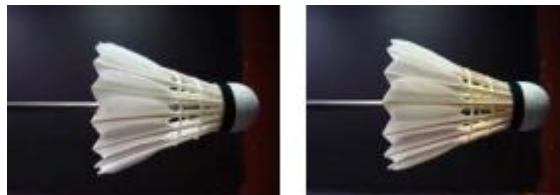


Fig.30: Synthetic and natural feather shuttlecock [Alam et. al, 2009]



Fig.31: Complete experimental setup [Alam et. al, 2009]

Try

1. Investigate the drag of natural feather shuttlecock (Fig. 32).
2. Compute the drag of synthetic feather shuttlecock (Fig. 31).
3. Examine the drag of all feather shuttlecocks (Fig. 30).

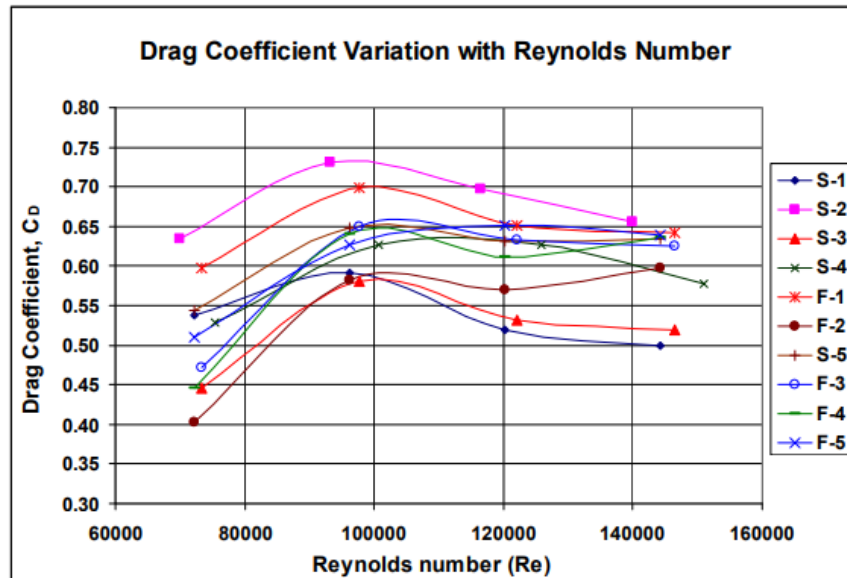


Fig.32: Drag coefficient variation with Reynolds number for all Shuttlecocks

V. TEXT BOOKS:

1. Alan pope, "Low Speed Wind Tunnel Testing", John Wiley, 2nd edition, 1999.

VI. REFERENCE BOOKS:

2. Mattingly J.D., "Elements of Propulsion: Gas Turbines and Rocket", AIAA, 1991.

VII. ELECTRONICS RESOURCES:

1. www.loc.gov/rr/scitech/tracer-bullets/aerodynamicstb.html
2. www.myopencourses.com/subject/aerodynamics-2

VIII. MATERIALS ONLINE

1. Course template
2. Lab manual



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

AEROSPACE MATERIALS AND PRODUCTION TECHNOLOGY LABORATORY								
IV Semester: AE								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AAED13	Core	L	T	P	C	CIA	SEE	Total
		-	-	2	1	40	60	100
Contact Classes: Nil	Tutorial Classes: Nil	Practical Classes: 45			Total Classes: 45			
Prerequisite: Aircraft Materials and Production								

I. COURSE OVERVIEW:

The Aircraft Production Technology lab encompasses on providing sound practical knowledge on testing of engineering material and conventional machining process which plays a vital role in designing the components with minimum cost and with longer service.

II. COURSE OBJECTIVES:

The students will try to learn:

- The basic material properties to identify the suitable applications in aerospace industries.
- The conventional machining techniques required for aircraft production.
- The tooling and material joining technique used in aircraft assembly.

III. COURSE OUTCOMES:

After successful completion of the course, students should be able to:

- CO1 Identify the microstructures of the materials for selecting the suitability in industrial applications
- CO2 Illustrate various jobs for joining the materials using welding operation in real time applications.
- CO3 Identify the types of machining process required for producing desired shape of components used in Aerospace and allied industries.
- CO4 Demonstrate molding processes and their application for producing machine components used in industries.
- CO5 Select the suitable tools and process parameters required in machining, drilling and slotting operations for producing components with minimum cost
- CO6 Illustrate various jobs for joining the materials using Riveting operation in industries.

IV. COURSE CONTENT

EXERCISES FOR AIRCRAFT PRODUCTION TECHNOLOGY LABORATORY

1. Getting Started Exercises

1.2 Introduction to Aircraft Production Technology Laboratory

- Understand the working principle of microscope, lathe and milling machines used in the laboratory.
- Become familiar with grinding, shaping, slotting and drilling machines.
- Learn to take quality control measures to ensure precision and accuracy in aircraft component fabrication.

Try

1. Prepare the aluminum sample for evaluation of microstructures
2. Plain turning operation on lathe machine for a given sample

2. Exercises on Basic Metallurgy -I

2.1 Microstructure of pure materials

Observe the micro structural features using the microscope shown in Figure 1 for Mild steel Specimen.



Figure 1 Microscope

Try

1. Change the specimen to Aluminum with different surface finish and repeat the same experiment
2. Change the sample to Mild Steel with different surface finish and repeat the same experiment
3. Perform the experiment on Copper with different surface finish and repeat the same experiment

3. Exercises on Basic Metallurgy -II

3.1 Microstructure of non-ferrous alloys

Find the Micro structural Properties of the materials like weight, strength, toughness, hardness, corrosion, fatigue resistance, performance in temperature extremes using the microscope as shown in Figure 2.

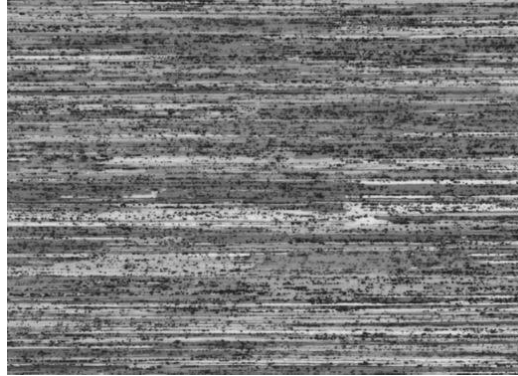


Figure 2 Microscopic view of MS plate

Try

1. Change the specimen to mild steel and find the above properties
2. Perform the experiment on Aluminum and find the above properties
3. Change the specimen to Steel and find the above properties

4. Exercises on Lathe Operations - I

4.1 Plane turning

Using the lathe Machine, conduct the plain turning operation for the given work piece (mild steel) as shown in Figure 3.

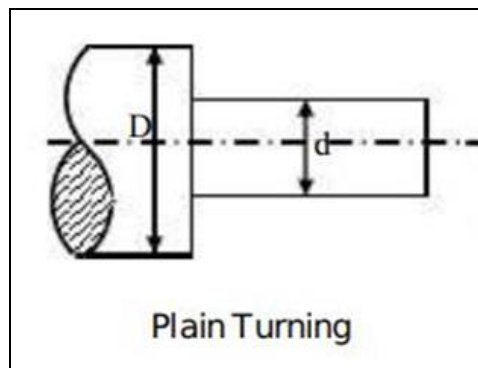


Figure 3 plain turning

4.2 Step turning and Grooving

Using the lathe Machine, conduct the step turning operation for the given work piece (mild steel) as shown in Figure 4.

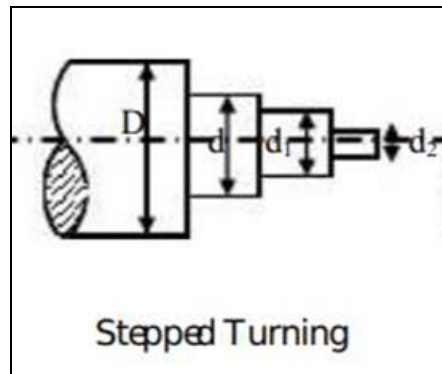


Figure 4 stepped turning

Try

1. Perform the Grooving operation for the Mild Steel(MS)Specimen.
2. Change the Specimen to steel the conduct the process.
3. Perform the step turning & grooving operation on Cast Iron Specimen.

5. Exercises on Lathe Operations - II

5.1 Taper turning-compound rest/offset method

Using the Lathe Machine, as shown in Figure 5 perform the Taper turning Operation for the given Specimen (Mild Steel)

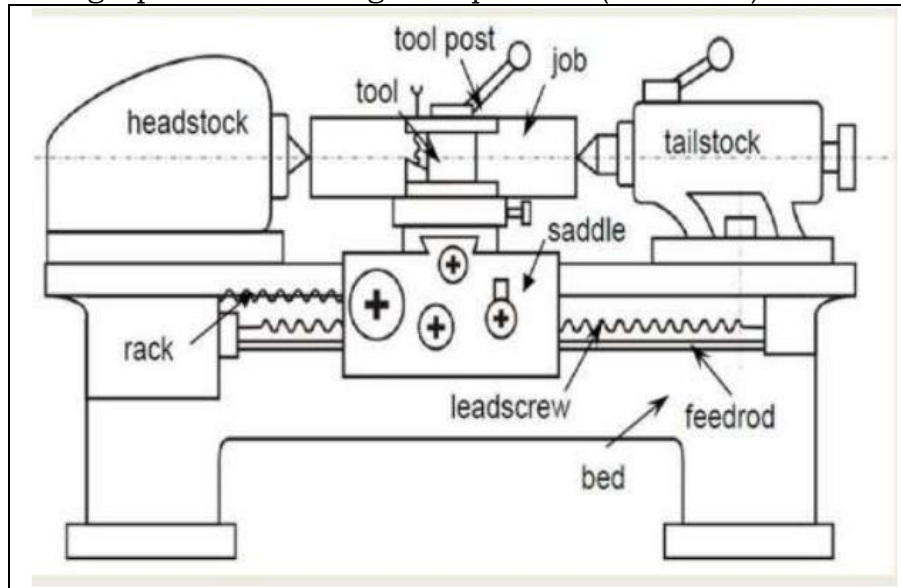


Figure 5 lathe machine

5.2 Drilling using lathe

Perform the drilling Operation on the Mild Steel work piece as shown in the Figure 6.

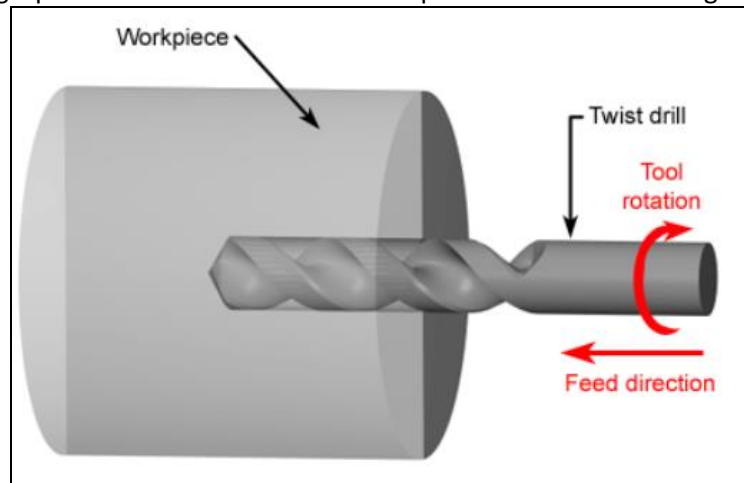


Figure 6 Drill bit & work piece

5.3 External threading-Single start

Perform the external threading operation on the mild steel work piece as shown in Figure 7.

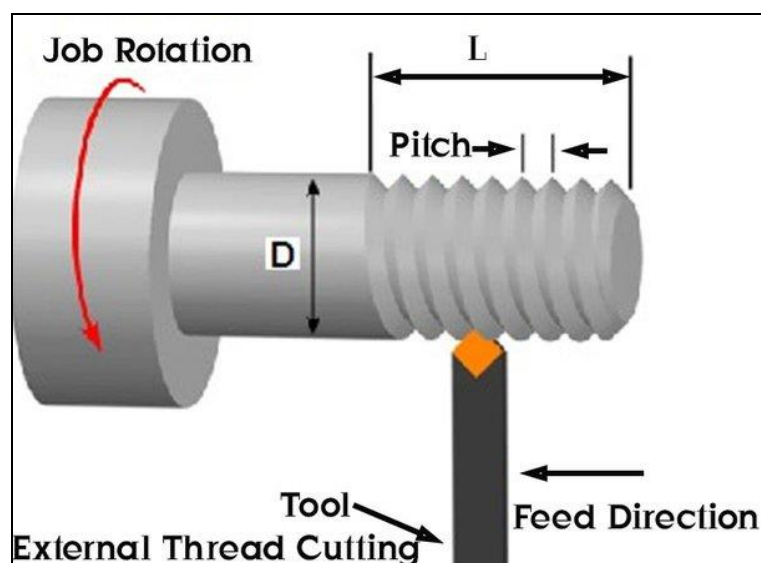


Figure 7 External threading operation by lathe

Try

1. Change the material to steel rod and perform the same operations.
2. Change the material to copper alloy and perform the threading operation.

6. Exercises on Shaping Machine

6.1 Shaping-V-Block

Using the Shaping Machine, perform the V- Shaped groove on Mild Steel as shown in Figure 8.

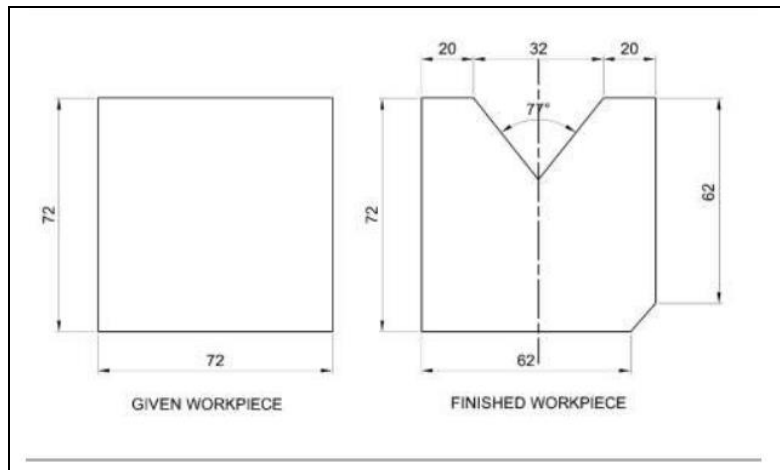


Figure 8 V-block shaping

Try

1. Change the Block to T Shape and perform the same operation on Mild Steel.
2. Perform the U shaped operation on the Mild steel work piece.
3. Change the tool bit to I Shape and perform the same operation on Mild Steel.

7. Exercises on Slotting Machine

7.1 Slotting-Keyways

Perform the Slotting Operation on the Mild Steel Specimen and obtain the required groove as shown in Figure 9b.

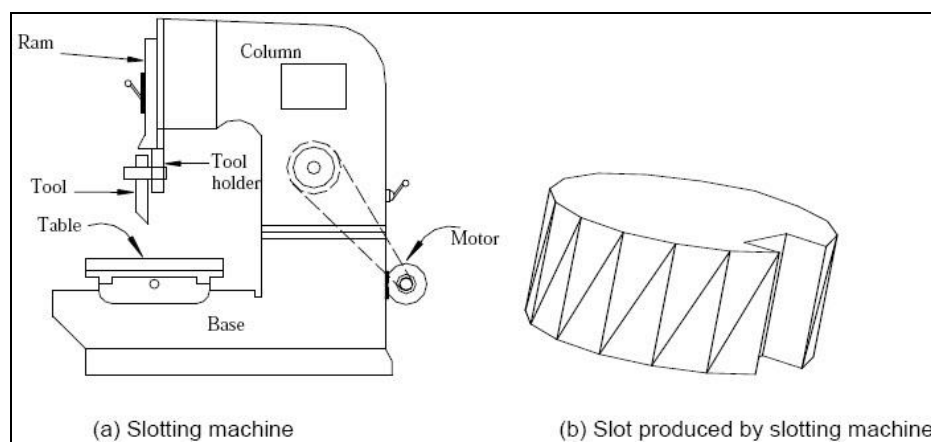


Figure 9 slotting machine & slot on work piece

Try

1. Slot the Cast Iron work piece and perform the same experiment.
2. Perform the Slotting Operation on the galvanized iron work piece and conduct the experiment.
3. Change the specimen to Steel with different surface finish and repeat the same experiment.

8. Exercises on Milling Machine

8.1 Milling-Face milling

Using the milling machine, perform the Face Milling Operation as shown in Figure 10, on the mild steel work piece as shown in Figure 11.

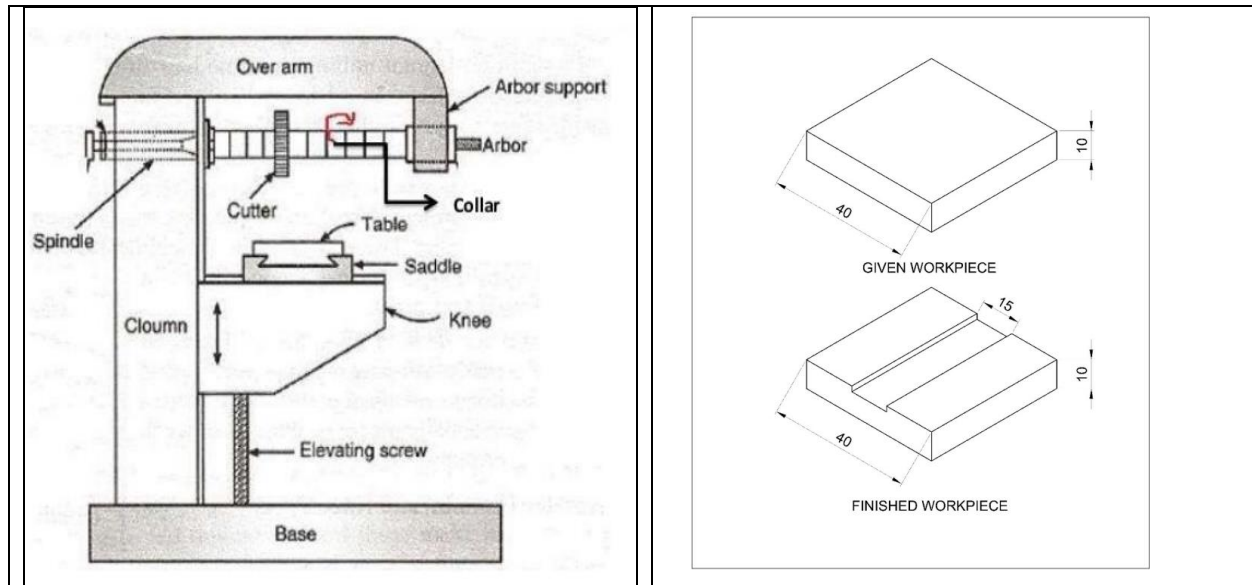


Figure 10 milling machine

Figure 11 work piece performed

8.2 End milling

Perform the End milling operation on given work piece as shown in the Figure 12.

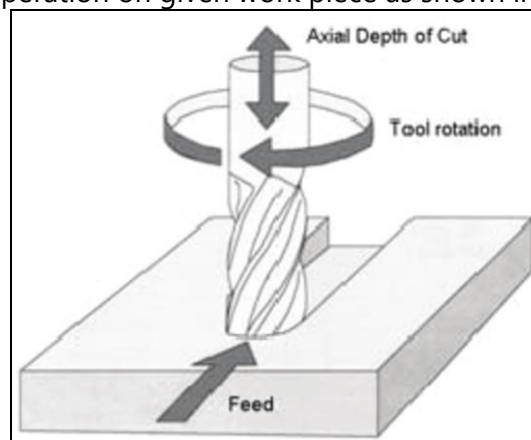


Figure 12 end milling performed

8.3 Side milling

Perform the side milling operation on the given work piece as shown in Figure 13.

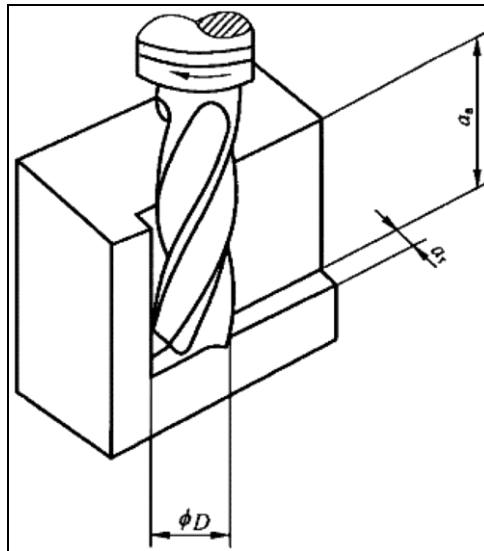


Figure 13 side milling performed

Try

1. Change the Specimen to steel and perform the same experiment.
2. Perform the end milling operation on the galvanized iron specimen.

9. Exercises on Grinding Machine

9.1 Grinding-Cylindrical /Surface/Tool & cutter.

Perform the Grinding Operation using grinding machine as shown in Figure 14, for the Mild Steel Specimen and remove the surface by 0.1 mm only as shown in the Figure 15.

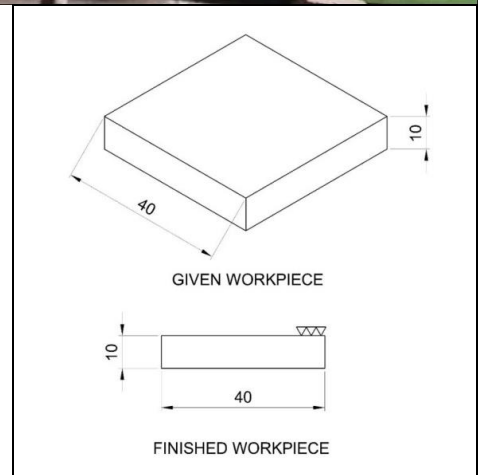


Figure 14 grinding machine

Figure 15 workpiece

Try

1. Grind the surface of Cast Iron and perform the same experiment.
2. Grind the surface of Galvanized Iron and repeat the same experiment.
3. Smoothen the surface of steel and redo the same experiment.

10. Exercises on Drilling Machine

10.1 Drilling, reaming Operation

Perform the Drilling & Reaming Operation on the Mild Steel Work Piece as shown in the Figure 16.

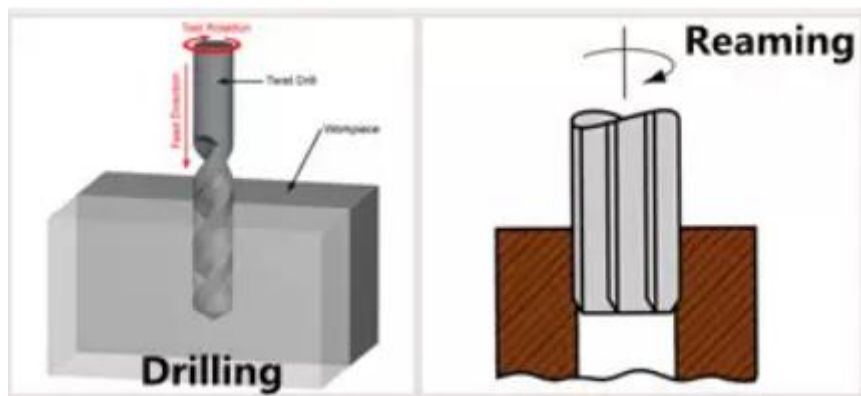


Figure 16 drilling & reaming

10.2 Counter boring, Counter sinking Operation

Perform the counter boring and counter sinking operation on the given specimen as shown in the Figure 17.

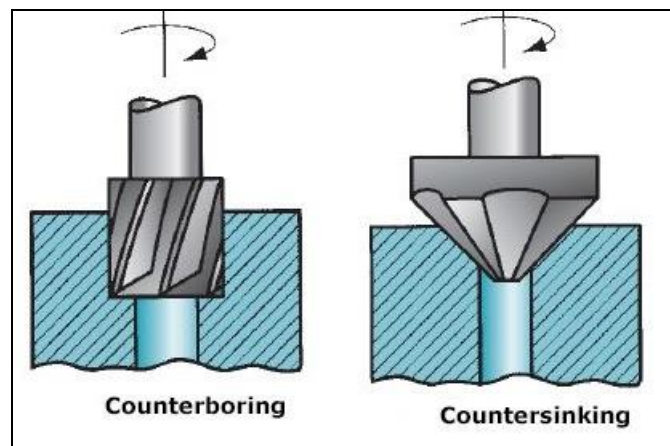


Figure 17 counter boring & counter sinking

Try

1. Perform the tapping operation in the mild steel specimen using the drilling machine.
2. Perform this reaming and counter boring operation on steel using the drilling machine.
3. Perform the counter sinking operation in the mild steel using drilling machine.

11. Exercises on Welding Process-I

11.1 Gas Welding

Using Gas welding as shown in Figure 18, perform the Butt Joint of two Mild steel Metal as shown in Figure 19.

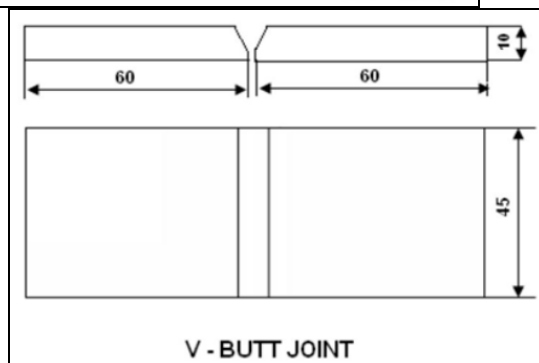
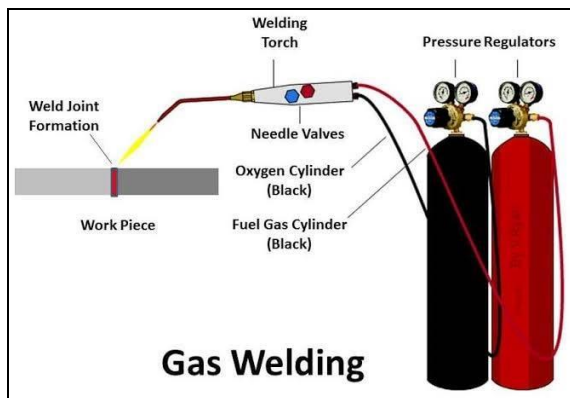


Figure 18 Gas welding

Figure 19 V- Butt joint

11.2 Brazing

Using brazing operation, perform the Joining of two Mild steel Metal pieces in Butt Joint as shown in Figure 20.

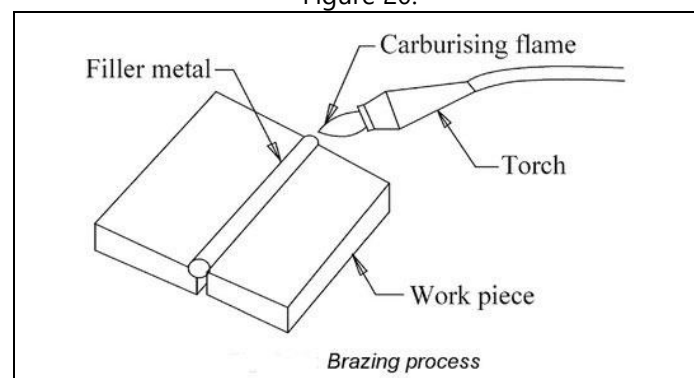


Figure 20 Brazing Operation

11.3 Soldering

using soldering, perform the Joining of two Mild steel Metal pieces in Butt Joint as shown in Figure 21.

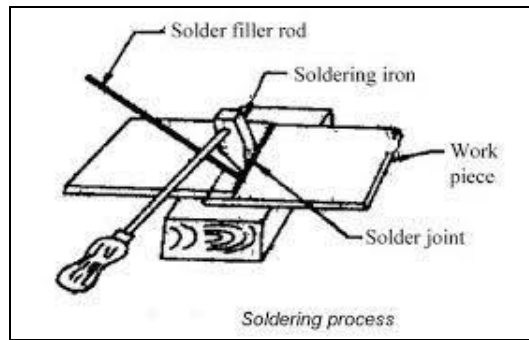


Figure 21 Soldering process

Try

1. Perform the joining process in steel sheets using the brazing operation.
2. Perform the joining process in the galvanized iron using the gas welding process.
3. Perform the joining process in the copper alloy using the gas welding & brazing process.

12. Exercises on Welding Process-II

12.1 Arc welding

Perform the Joining of two Mild Steel Work pieces using Arc welding as shown in Figure 22.

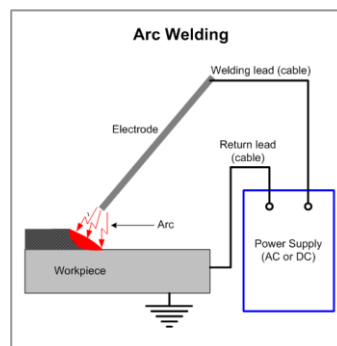


Figure 22 Arc welding

12.2 Spot welding

Perform the Joining of two Mild Steel Work pieces using the Spot Welding as shown in Figure 23.

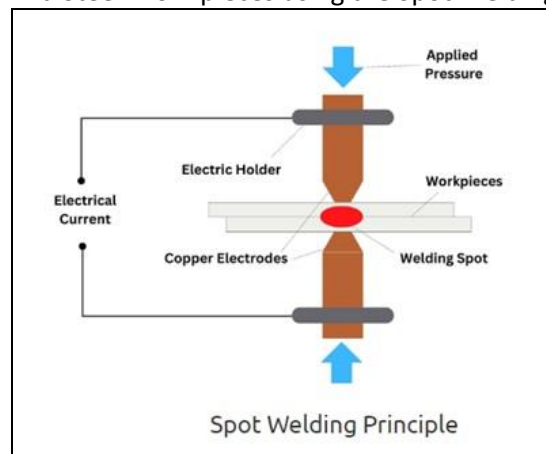


Figure 23 Spot welding principle

Try

1. Perform the Spot welding Operation using fixed length and find the strength.
2. Perform the Spot welding Operation Using Variable spot welding length and find the strength.
3. Perform the Arc welding operation with aluminum & riveting joint.

13. Exercises on Basic Casting

13.1 Preparation of casting with simple patterns.

Create a new solid rod with 50 mm length and 10 mm radius from the mold using the casting operation as shown in Figure 24.

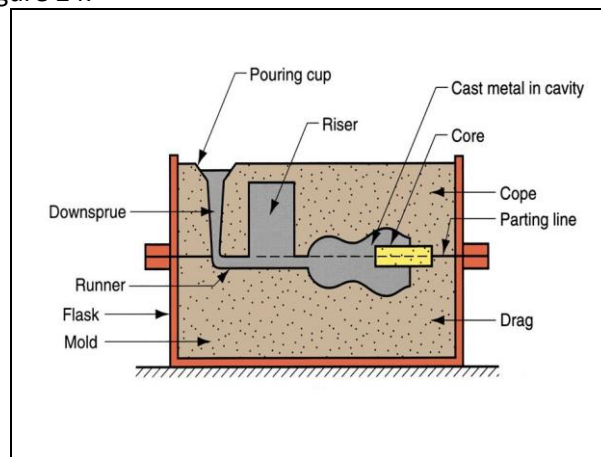


Figure 24 Basic Casting setup

Try

1. Create a Dumbbell using the casting operation.
2. Create a solid chair leg using the casting process.
3. Create a step based rod of aluminum material using the casting process.

13. Exercises on Injection Molding

13.1 Blow Molding

Perform the molding process using blow molding and make a new model out of it as shown in Figure 25.

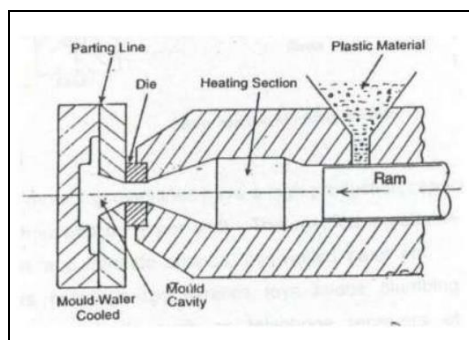


Figure 25 Blow molding

13.2 Injection Molding

Perform the molding process using Injection Molding and make a new model out of it as shown in Figure 26.

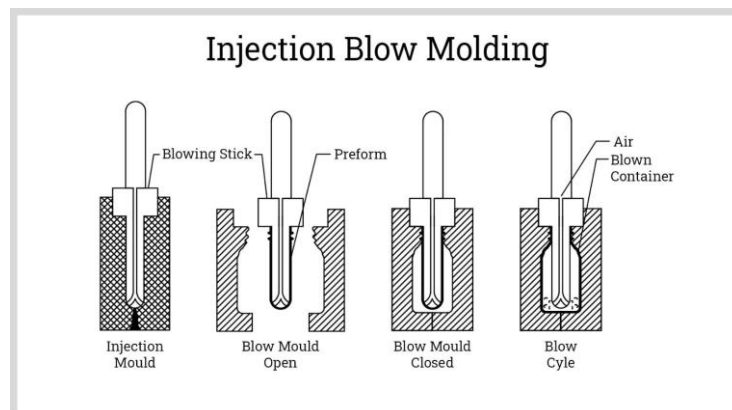


Figure 26 Injection Blow molding

Try

1. Create a cylindrical mold of Aluminum using the above two process.
2. Create a cylindrical mold of copper rod using the two processes mentioned above.
3. Create a dumbbell rod with 2.5 kg weight of aluminum metal.

14. Exercises on Additive Manufacturing

14.1 Cold riveting

Perform the Joining Process using the Riveting operation and find the strength as shown in the Figure 27.

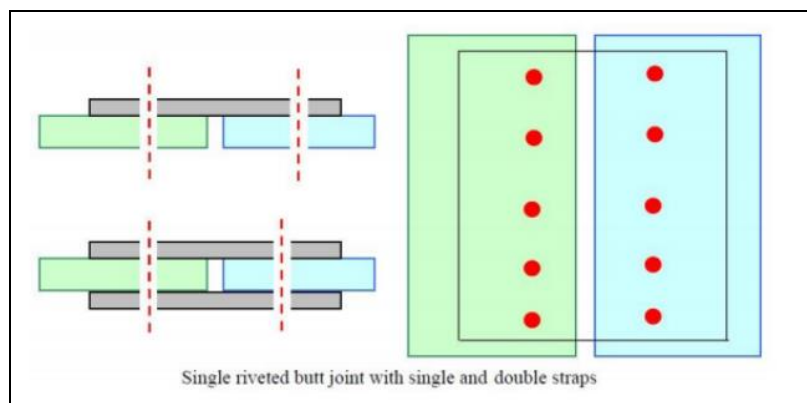


Figure 27 Single riveted butt joint with single & double straps

14.2 Hot riveting

Perform the Joining Process using the hot riveting operation where rivets are initially heated before applying force as shown in the Figure 28.

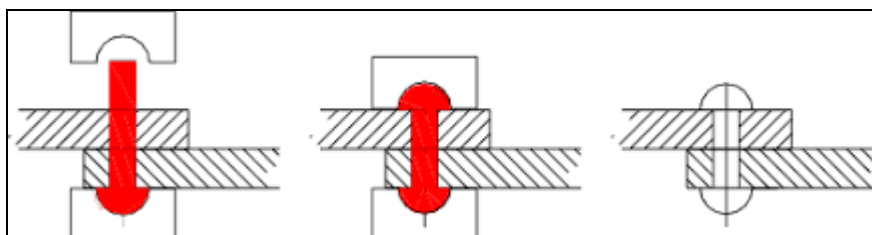


Figure 28 Hot riveting performed on work piece

Try

1. Perform the joining process using riveting of Aluminum Sheets with 5 rivets only.
2. Perform the Joining Process using Riveting of Galvanized sheets with 5 rivets only.
3. Perform the joining process using cold and hot riveting with 5 rivets only.

15. Exercises on Fused Deposition Modelling (FDM)

15.1 3D printing of a part

Perform the part modeling using 3D printing make a new model out of it as shown in Figure 29.

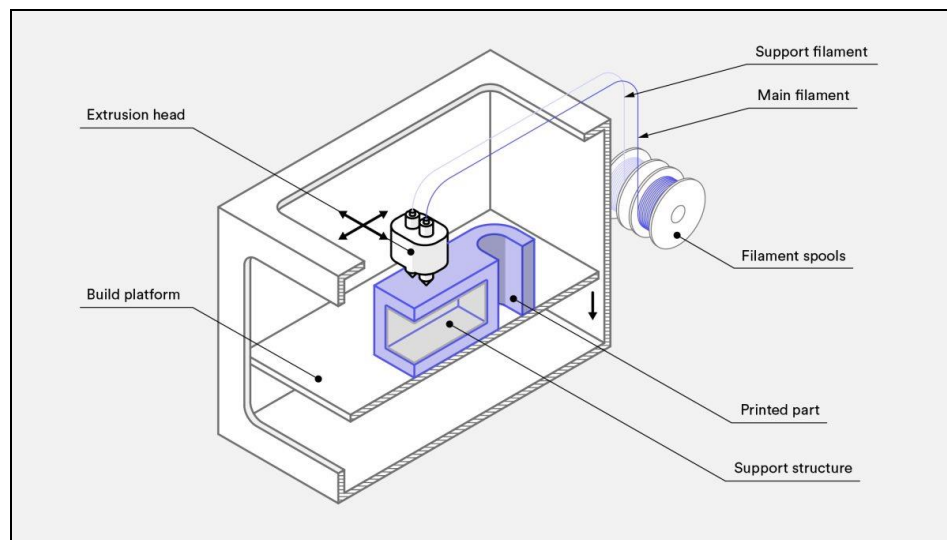


Figure 29 Part modeling in 3D printing

15.2 3D Printing of Gear

Perform the 3D modeling of a gear and make a new model out of it as shown in Figure 30.



Figure 30 3D printing of Gear

Try

1. Create a aircraft brackets by 3D modeling.
2. Create a aircraft wing modeling.
3. Create the turbine blades of the RC aircraft.

V. TEXT BOOKS:

1. Keshu S. C, Ganapathy K. K, “Air craft production techniques”, Interline Publishing House, Bangalore, 3rd Edition, 1993.

VI. REFERENCE BOOKS:

1. R. K Jain-Khanna, “Production technology”, McGraw Hill, 1st edition, 2002.
2. O. P Khanna, Lal. M. DhanpatRai, “Production technology, 5th edition, 1997.
3. C. P. Paul, A. N. Jinoop, “Additive Manufacturing”, McGraw Hill India 1st edition, 2021,

VII. ELECTRONICS RESOURCES:

1. www.loc.gov/rr/scitech/tracer-bullets/aerodynamicstb.html
2. www.myopencourses.com/subject/aerodynamics-2

VIII. MATERIALS ONLINE

1. Course template
2. Lab manual



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

DevOps Engineering								
IV Semester: AERO ME CE EEE CSE IT CSE (AI&ML) CSE (DS) CSE (CS)								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
ACSD18	Skill	L	T	P	C	CIA	SEE	Total
		2	0	0	0	40	60	100
Contact Classes: 32	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: 32			
Prerequisite: OBJECT ORIENTED PROGRAMMING								

I. COURSE OVERVIEW:

DevOps, a combination of "development" and "operations," is a software development methodology that emphasizes collaboration and communication between software developers and IT operations professionals. The goal of DevOps is to streamline the software delivery process, from code development to deployment and maintenance, by breaking down silos between development and operations teams.

II. COURSES OBJECTIVES:

The students will try to learn

- The DevOps Concepts for business cases, cloud provisioning and management services .
- The model canvas for DevOps use cases.
- The virtual machines and containers for designing of applications
- The code with various aspects in continuous deployment / development.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO 1 Understands the DevOps concepts in continuous delivery / development of applications.
- CO 2 Create the DevOps applications using various tools and technologies.
- CO 3 Examine the virtual machines and containers for managing the files
- CO 4 Apply cloud services for deployment the applications in a real-time
- CO 5 Perform web security and testing the code with appropriate tools

IV. COURSE SYLLABUS:

MODULE 1: DevOps Concepts

Understanding DevOps movement, DevOps with changing time, The water fall model, Agile Model, Collaboration, Why DevOps, Benefits of DevOps, DevOps life cycle- all about continuous, Build Automation, Continuous Integration, Continuous Management, Continuous Delivery / Continuous Development, The agile wheel of wheels.

MODULE 2: DevOps Tools and Technologies

Code Repositories : Git, Differences between SVN and Git, Build tools – Maven, Continuous integration tools – Jenkins, Container Technology – Docker, Monitoring Tools, Continuous integration with Jenkins 2, Creating built-in delivery pipelines, Creating Scripts, Creating a pipeline for compiling and executing test units, Using the Build Pipeline plugin, Integrating the deployment operation

MODULE 3: Docker Containers

Overview of Docker containers, Understanding the difference between virtual machines and containers, Installation and configuration of Docker, Creating your first Docker container, Managing containers, Creating a Docker image from Docker file, An overview of Docker's elements, Creating a Dockerfile, Writing a Dockerfile,

Building and running a container on a local machine, Testing a container locally, Pushing an image to Docker Hub

MODULE 4: Cloud Provisioning and Configuration Management with Chef, Managing Containers Effectively with Kubernetes

Amazon EC2, Creating and configuring a virtual machine in Amazon Web Services, Prerequisite – deploying our application on a remote server, Deploying the application on AWS, Deploying the application in a Docker container.

Kubernetes architecture overview, Installing Kubernetes on a local machine, Installing the Kubernetes dashboard, Kubernetes application deployment, Using Azure Kubernetes Service (AKS), Creating an AKS service, Configuring kubectl for AKS, The build and push of the image in the Docker Hub

MODULE 5: Testing the Code

Manual testing, Unit testing, JUnit in general and JUnit in particular, A JUnit example, Automated integration testing, Docker in automated testing, Performance testing, Automated acceptance testing, Automated GUI testing, Integrating Selenium tests in Jenkins, JavaScript testing, Testing backend integration points, Test-driven development, A complete test automation scenario, Manually testing our web application.

V. TEXT BOOKS:

1. Mitesh Soni, “DevOps for Web Development”, Packt Publishing, 2016.
2. Mikael Krief, “Learning DevOps - The complete guide to accelerate collaboration with Jenkins, Kubernetes, Terraform and Azure DevOps”, Packt Publishing, 2019.

VI. REFERENCE BOOKS:

1. Joakim Verona, “Practical DevOps”, Packt Publishing, 2016.
2. Michael Huttermann, “DevOps for Developers”, A press publishers, 2012.
3. Sanjeev Sharma, “The DevOps Adoption Playbook”, Published by John Wiley & Sons, Inc.2017.
4. Sanjeev Sharma & Bernie Coyne, “DevOps for Dummies”, Published by John Wiley & Sons, Inc.

VII. REFERENCE BOOKS:

1. <https://www.geeksforgeeks.org/devops-tutorial/>
2. <https://www.javatpoint.com/devops>
3. <https://azure.microsoft.com/en-in/solutions/devops/tutorial>
4. <https://www.guru99.com/devops-tutorial.html>



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

UNDERTAKING BY STUDENT / PARENT

“To make the students attend the classes regularly from the first day of starting of classes and be aware of the College regulations, the following undertaking form is introduced which should be signed by both student and parent. The same should be submitted to the Dean of Academic”.

I, Mr. / Ms. ----- joining I Semester / III Semester for the academic year 20 - 20 / 20 - 20 in Institute of Aeronautical Engineering, Hyderabad, do hereby undertake and abide by the following terms, and I will bring the ACKNOWLEDGEMENT duly signed by me and my parent and submit it to the Dean of Academic.

1. I will attend all the classes as per the timetable from the starting day of the semester specified in the institute Academic Calendar. In case, I do not turn up even after two weeks of starting of classes, I shall be ineligible to continue for the current academic year.
2. I will be regular and punctual to all the classes (theory/laboratory/project) and secure attendance of not less than 75% in every course as stipulated by Institute. I am fully aware that an attendance of less than 65% in more than 60% of theory courses in a semester will make me lose one year.
3. I will compulsorily follow the dress code prescribed by the college.
4. I will conduct myself in a highly disciplined and decent manner both inside the classroom and on campus, failing which suitable action may be taken against me as per the rules and regulations of the institute.
5. I will concentrate on my studies without wasting time in the Campus/Hostel/Residence and attend all the tests to secure more than the minimum prescribed Class/Sessional Marks in each course. I will submit the assignments given in time to improve my performance.
6. I will not use Mobile Phone in the institute premises and also, I will not involve in any form of ragging inside or outside the campus. I am fully aware that using mobile phone to the institute premises is not permissible and involving in Ragging is an offence and punishable as per JNTUH/UGC rules and the law.
7. I declare that I shall not indulge in ragging, eve-teasing, smoking, consuming alcohol drug abuse or any other anti-social activity in the college premises, hostel, on educational tours, industrial visits or elsewhere.
8. I will pay tuition fees, examination fees and any other dues within the stipulated time as required by the Institution / authorities, failing which I will not be permitted to attend the classes.
9. I will not cause or involve in any sort of violence or disturbance both within and outside the college campus.
10. If I absent myself continuously for 3 days, my parents will have to meet the HOD concerned / Principal.
11. I hereby acknowledge that I have received a copy of BT23 Academic Rules and Regulations, course catalogue and syllabus copy and hence, I shall abide by all the rules specified in it.

ACKNOWLEDGEMENT

I have carefully gone through the terms of the undertaking mentioned above and I understand that following these are for my/his/her own benefit and improvement. I also understand that if I/he/she fail to comply with these terms, shall be liable for suitable action as per Institute/JNTUH/AICTE/UGC rules and the law. I undertake that I/he/she will strictly follow the above terms.

Signature of Student with Date

**Signature of Parent with Date
Name & Address with Phone Number**