UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

Lecture-wise Plan

Subject Name: Mathematics-III **Subject Code-M301**
Year: 2nd Year **Semester: Third**

Module Number	Topics	Number of Lectures
1	**Fourier Series & Fourier Transform**	**8L**
	Introduction	1
	Fourier Series for functions of period 2π, Fourier Series for functions of period 2L	3
	Fourier Integral Theorem, Fourier Transform of a function, Fourier Sine and Cosine Integral Theorem.	1
	Properties of Fourier Transform, Fourier Transform of Derivatives.	1
	Convolution Theorem, Inverse of Fourier Transform.	2
2	**Introduction to Functions of a Complex Variable & Conformal Mapping, Complex Integration, Residue & Counter Integration**	**8L**
	Complex functions, Limit, Continuity and Differentiability, Analytic functions	1
	Cauchy-Riemann Equations, Harmonic function and Conjugate Harmonic function	1
	Construction of Analytic functions: Milne Thomson method.	1
	Simple curve, closed curve, smooth curve & contour, complex Integrals.	1
	Cauchy's theorem, Cauchy-Goursat theorem, Cauchy's integral formula, Cauchy's integral formula	2
3	**Basic Probability Theory, Random Variable & Probability Distributions. Expectation**	**12L**
	Introduction	1
	Conditional probability, Independent events & Multiplication Rule.	1
	Baye's theorem	1
	Random variable	1
	Probability density function & probability mass function.	2
	Expectation & Variance	1
	Binomial & Poisson distributions and related problems.	2
	Uniform, Exponential, Normal distributions and related problems.	3
4	**Partial Differential Equation (PDE) and Series solution of Ordinary Differential Equation (ODE)**	**7L**
	Origin of PDE, its order and degree, concept of solution in PDE.	1
	Different methods: Separation of variables, Laplace & Fourier transform methods.	3
	PDE I: One dimensional Wave equation.	1
	PDE II: One dimensional Heat equation	1
	PDE III: Two dimensional Laplace equation.	1

Assignment:

Module-1:

1. Write the statement of Fourier integral Theorem.
2. If the Fourier series of function $f(x)$ is given by $a_0 + \sum_{n=1}^{\infty} a_n \cos nx + \sum_{n=1}^{\infty} b_n \sin nx$, then a_n is given by?
3. If the Fourier series of function $f(x)$ is given by $a_0 + \sum_{n=1}^{\infty} a_n \cos nx + \sum_{n=1}^{\infty} b_n \sin nx$, then b_n is given by?
4. If the Fourier series of function $f(x)$ is given by $a_0 + \sum_{n=1}^{\infty} a_n \cos nx + \sum_{n=1}^{\infty} b_n \sin nx$, then a_n is given by?
5. If the Fourier series of function $f(x)$ is given by $a_0 + \sum_{n=1}^{\infty} a_n \cos nx + \sum_{n=1}^{\infty} b_n \sin nx$, then b_n is given by?
6. If $F(p)$ is the Fourier transform of $f(x)$, then the Fourier transform of $f(ax)$ is given by?
7. If $F(p)$ is the Fourier transform of $f(x)$, then the Fourier transform of $f(x-a)$ is given by?
8. If $F(p)$ is the Fourier transform of $f(x)$, then the Fourier transform of $f(ax)$ is given by?
9. If $F(p)$ is the Fourier transform of $f(x)$, then the Fourier transform of $f(x-a)$ is given by?
10. Define periodic function
11. Define even function
12. Write the relation between two orthogonal functions.
13. IF convolution of two functions exists then the value of $F\left\{\frac{1}{\sqrt{2®}}\int_{-\infty}^{\infty} f(u)g(x-u)du; p\right\} =$
14. IF convolution of two functions exists then the value of $F\left\{\frac{1}{\sqrt{2®}}\int_{-\infty}^{\infty} f(u)g(x-u)du; p\right\} =$
15. IF convolution of two functions exists then the value of $F\left\{\frac{1}{\sqrt{2®}}\int_{-\infty}^{\infty} f(u)g(x-u)du; p\right\} =$
16. IF convolution of two functions exists then the value of $F\left\{\frac{1}{\sqrt{2®}}\int_{-\infty}^{\infty} f(u)g(x-u)du; p\right\} =$
17. Obtain the Fourier series for the function $f(x) = x^2, -® < x < ®$.
18. Obtain the fourier series for the function $f(x) = \frac{1}{4}(® - x^2), 0 < x < 2®$.
19. Obtain the fourier series for the function $f(x) = \sin ax, -® < x < ®$. a being non-integer value.
20. Obtain the fourier series for the function $f(x) = x, -® < x < ®$.

Module-2:

21. Write Cauchy- Riemann equations for a function $f(z) = u(x,y) + iv(x,y)$.

UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

Lecture-wise Plan

22. Write necessary condition for a function $f(z)=u(x,y)+iv(x,y)$ to be analytic.
23. Write necessary and sufficient condition for a function $f(z)=u(x,y)+iv(x,y)$ to be analytic.
24. Write sufficient condition for a function $f(z)=u(x,y)+iv(x,y)$ to be analytic.
25. State Cauchy's integral theorem.
26. Write Cauchy's integral formula.
27. Write type of singularity of the function $\frac{\sin z}{z}$ at $z=0.$
28. Write type of singularity of the function $\frac{z^3}{(z+1)^2(z-5)^4}$ at $z=5.$
29. Write type of singularity of the function $\frac{1}{(z+1)^2(z-3)^2}$ at $z=-1.$
30. Write type of singularity of the function $\frac{z^2}{(z+1)(z-3)^2}$ at $z=3.$
31. Examine that the function $f(x,y)=y^3-3x^2y$ is harmonic or not.
32. Examine that the function $f(x,y)=\frac{1}{2}\log(x^2+y^2)$ is harmonic or not.
33. Examine that the function $f(x,y)=\frac{x-y}{x^2+y^2}$ is harmonic or not.
34. Examine that the function $f(x,y)=2x(1-y)$ is harmonic or not.
35. Evaluate $\int_0^{1+i} z^2dz$, where z is complex number.
36. Evaluate $\int_0^{1+2i}(1+z^2)dz$, where z is complex number.
37. Evaluate $\int_0^{2+i} e^z dz$, where z is complex number.
38. Evaluate $\int_0^{1+i}(z^2+3z+2)dz$, where z is complex number.
39. Find the residue at the poles of $f(z)=\frac{\cot \circledR z}{(z-a)^2}.$
40. Find the residue at the poles of $f(z)=\frac{z^2-2z}{(z+1)^2(z^2+4)}.$
41. Find the residue of $f(z)=\frac{z^3}{z^2-1}$ at $z=\infty.$
42. Find the residue of $f(z)=\frac{e^z}{z\sin mz}$ at $z=0.$

Module-3:

1. If for two events A and B we have the following probabilities:

$P(A)=P(A|B)=\frac{1}{4}; P(B|A)=\frac{1}{2}$. Then check A and B are independent or not.

2. If $P(A\cap B)=\frac{1}{2}, P(\overline{A}\cap\overline{B})=\frac{1}{2}$ and $2P(A)=P(B)=p$, then find the value of p.
3. If for two events A and B we have the following probabilities:

   $P(A)=P(A|B)=\frac{1}{4}; P(B|A)=\frac{1}{2}$. Then find $P(\overline{A}|B)=$.
4. If $P(A\cap B)=\frac{1}{2}, P(\overline{A}\cap\overline{B})=\frac{1}{3}$ and $P(A)=P(B)=p$, then find the value of p.
5. If A and B are any two events and $P(A)=p_1; P(B)=p_2; P(A\cap B)=p_3$. Then $P(\overline{A}\cup\overline{B})=$
6. If A and B are any two events and $P(A)=p_1; P(B)=p_2; P(A\cap B)=p_3$. Then $P(\overline{A\cup B})=$
7. If A and B are any two events and $P(A)=p_1; P(B)=p_2; P(A\cap B)=p_3$. Then $P(\overline{A}\cap\overline{B})=$
8. If A and B are any two events and $P(A)=p_1; P(B)=p_2; P(A\cap B)=p_3$. Then $P(\overline{A}\cup B)=$
9. State Baye's theorem for mutually disjoint events.
10. If $f(x)=\begin{cases} ke^{-2x} & x\geq 0 \\ 0 & otherwise \end{cases}$, then what will be the value of k for which $f(x)$ be probability density function?
11. If $f(x)=\begin{cases} x & 0<x<1 \\ 0 & otherwise \end{cases}$, then $f(x)$ is probability density function or not?
12. If $f(x)=\begin{cases} ke^{-x} & x\geq 0 \\ 0 & otherwise \end{cases}$, then what will be the value of k for which $f(x)$ be probability density function?
13. If $f(x)=\begin{cases} k\left(1-e^{-x}\right)^2 & x\geq 0 \\ 0 & otherwise \end{cases}$, then what will be the value of k for which $f(x)$ be probability density function?
14. Write the formula for mathematical expectation of a discrete random variable X with probability mass function $f(x)$.
15. Write the formula for mathematical expectation of a continuous random variable X with probability density function $f(x)$.
16. Write the formula for mathematical expectation of a discrete random variable X with probability mass function $f(x)$.
17. Write the formula for mathematical expectation of a continuous random variable X with probability density function $f(x)$.

18. A card is drawn from pack of 52 cards, find the probability of getting a king or a heart or a red card?
19. A card is drawn from a pack of 52 cards, if the value of faces cards 10, aces cards 1 and other according to denomination, find the expected value of the no. of point on the card.
20. A bag contains 10 red and 15 white balls. Two balls are drawn in succession. What is the probability that one of them is white and other red?
21. State Bayes' theorem.
22. A and B take turns in throwing two dice on the understanding that the first to throw 9 will be awarded a prize. If A has the first turn, show their respective chances of winning are in the ratio 9 : 8.
23. Three groups of children contain respectively 3 girls and 1boy; 2 girls and 2 boys; 1 girls and 3 boys, One child is selected at random from each group. Find the chance of selecting 1 girl and 2 boys.
24. A manufacturer supplies quarter horsepower motors in lots of 25. A buyer, before taking a lot, tests at random a sample of 5 motors and accepts the lot if they are all good; otherwise he rejects the lot. Find the probability that :(i) he will accept a lot containing 5defective motors ;(ii) he will reject a lot containing only one defective motors.
25. In an examination with multiple-choice questions, each question has four, out of which one is correct. A candidate ticked the answer either by his skill or by copying from his neighbours, The probability of guess is 1/3, copying is 1/6. The probability of correct answer by copying is 1/8. If a candidate answers a question correctly find the probability that he know the answer.
26. An urn contains 10 white and 3 black balls. Another urn contains 3 white and 5 black balls. Two balls are drawn at random from first urn and placed in the second urn and then one ball is taken at random from the latter. What is the probability that it is a white ball ?
27. Define the random variable, Explain the types of random variable with example.
28. A can hit a target 4 times in 7 shots, B 3 times in 5 shots and C three times in 5 shots. All of them fire one shot each simultaneously at the target. What is the probability that (i) 2 shots hit (ii) At least two shots hit ?
29. The probability that a student A solves a mathematics problem is 2/5 and the probability that a student B solves the problem is 2/3. What is the probability that (a) the problem is not solved (b) the problem is solved (c) both A and B solve the problem.
30. A company has four production section S_1, S_2, S_3& S_4 which contribute 30%, 20% 22% & 28% respectively produced 1%, 2%, 3% & 4% defective units, if a small unit is selected random & found to be defective, what is the probability that the unit selcected has came from (a) Section S_1 (b) Section S_4
31. From a city population, the probability of selecting a male or a smoker is 7/10, a male smoker is 2/5 and a male if a smoker is already selected is 2/3, find the probability of selecting (a) non-smoker (b) a male (c) a smoker if a male is first selected.
32. There are two bags A and B. A contains n white and 2 black balls & B contains 2 white and n black balls, one of the two bags is selected at random and two balls are drawn from it without replacement. If the both balls are drawn are white and the probability that the bag A was used to drawn the ball is 6/7. Find the value of n.

Module-4:

1. Bessel function of order $p = \pm\frac{1}{2}$, show that $J_{1/2}(x) = \sqrt{2/®x}\sin x$ and $J_{-1/2}(x) = \sqrt{2/®x}\cos x$.
2. Determine the order p of the following Bessel equation:
   a) $x^2y^{"} + xy' + (x^2 - 9)y = 0$

b) $x^2 y'' + xy' + x^2 y = 0$

3. Solve the following heat flow problem:

$$\frac{\partial u}{\partial t} = 7\frac{\partial^2 u}{\partial x^2} \quad , \quad 0 < x < ® \quad , \quad t > 0.$$

$$u(0,t) = u(®\ ,t) = 0\ ,\ \ t > 0\ ,$$

$$u(x,0) = 3\sin 2x - 6\sin 5x \ \ , \ 0 < x < \pi.$$

4. Prove that F satisfies the Laplace's equation: $F = Cz^n$

$$\nabla^2 F = \frac{\partial^2 F}{\partial x^2} + \frac{\partial^2 F}{\partial y^2} = 0$$

UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

Lecture-wise Plan

Subject Name: Data Structure & Algorithm **Subject Code-CS(CE)301**
Year: 2nd Year **Semester: Third**

Module Number	Topics	Number of Lectures
1	**Introduction:**	**5L**
	1. Why we need data structure? Concepts of data structures: a) Data and data structure b) Abstract Data Type and Data Type. Algorithms and programs, basic idea of pseudo-code. .	1
	2. Algorithm efficiency and analysis, time and space analysis of algorithms – order notations.	4
	Linear data structure:	
2	**Array:**	**2L**
	1. Different representations – row major, column major. Sparse matrix - its application and usage. Array representation of polynomials.	2
3	**Linked List:**	**7L**
	1. Singly linked list, circular linked list, doubly linked list, linked list representation of polynomial and applications.	7
4	**Stack and Queue:**	**6L**
	1. Stack and its implementations (using array, using linked list), applications.	2
	2. Queue, circular queue, dequeues. Implementation of queue- both linear and circular (using array, using linked list), applications.	4
5	**Recursion:**	**3L**
	1. Principles of recursion – use of stack, differences between recursion and iteration, tail recursion.	1
	2. Applications - The Tower of Hanoi, Eight Queens Puzzle.	2
	Non Linear data structure:	
6	**Trees:**	**8L**
	1. Basic terminologies, forest, tree representation (using array, using linked list). Binary trees - binary tree traversal (pre-, in-, post- order), threaded binary tree (left, right, full) - non-recursive traversal algorithms using threaded binary tree, expression tree.	4
	2. Binary search tree- operations (creation, insertion, deletion, searching). Height balanced binary tree – AVL tree (insertion, deletion with examples only). B- Trees – operations (insertion, deletion with examples only	4
	Graphs:	**5L**
	1. Graph definitions and concepts (directed/undirected graph, weighted/un-weighted edges, sub-graph, degree, cut-	

7	vertex/articulation point, pendant node, clique, complete graph, connected components – strongly connected component, weakly connected component, path, shortest path, isomorphism). Graph representations/storage implementations – adjacency matrix, adjacency list, adjacency multi-list.	1
	2. Graph traversal and connectivity – Depth-first search (DFS), Breadth-first search (BFS) – concepts of edges used in DFS and BFS (tree-edge, back-edge, cross-edge, forward-edge), applications.	2
	3. Minimal spanning tree – Prim's, Kruskal and Dijkstra algorithm (basic idea of greedy methods).	2
8	**Sorting, Searching and Hashing Technique:**	
	Sorting Algorithms:	6L
	Bubble sort and its optimizations, insertion sort, shell sort, selection sort, merge sort, quick sort, heap sort (concept of max heap, application – priority queue), radix sort.	6
	Searching:	2L
	Sequential search, binary search, interpolation search.	2
	Hashing:	2L
	Hashing functions, collision resolution techniques.	2
Total Number Of Hours = 46		

Faculty In-Charge HOD, CSE Dept.

Assignment:

Module-1(Introduction):

1. DefineAbstractData Type, big oh, big omega, theta notationoftimecomplexity.
2. Findthetotalfrequency count of following code.

```
for send=1ton do
        forreceive=1tosenddo
                forack=2toreceivedo
                        message=send-(receive+ack)
                        ack=ack-1
                        send=send+1
                end
        end
end
```

Module-2 (Linear data Structure):

1. Write a function to insert a element after 4^{th} position in an array.
2. Write a function to insert a element before 4^{th} position in a single linked list
3. Write a function to insert a element after a particular data element 4 in a doubly linked list.
4. Write a function to concatenate two circular linked list.

5. Write a function to implement stack and queue using linked list.
6. Convert infix to prefix and postfix.
   A+B+C-D/E*R(S*T)/W+G
7. Define tail and tree recursion, explain them with example.

Module-3(Non-linear data structure):

1. Why AVL tree is required?
2. Construct the AVL tree.
   B,D,A,G,H,R,J,T,C,Y,X
3. Write a short note on B-Tree.
4. Write an algorithm of DFS and Dijkstra algorithm.

Module-4(Sorting, Searching and Hashing):

1. Explain quick and radix sort with example.
2. Why binary search is better than linear search.
3. Write down different techniques of collision resolution techniques.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: **Basic Environmental Engineering** Subject Code: **CH-301**
Year: **2nd Year** Semester: **Third**

Module Number	Topics	Number of Lectures
1	**Chapter 1: General**	**6L**
	1. Basic ideas of environment, basic concepts, man, society & environment, their interrelationship.	1L
	2. Mathematics of population growth and associated problems, Importance of population study in environmental engineering, definition of resource, types of resource, renewable, non-renewable, potentially renewable, effect of excessive use vis-à-vis	2L
	3. Materials balance: Steady state conservation system, steady state system with non conservative pollutants, step function.	1L
	4. Environmental degradation: Natural environmental Hazards like Flood, earthquake, Landslide-causes, effects and control/management; Anthropogenic degradation like Acid rain-cause, effects and control. Nature and scope of Environmental Science and Engineering.	2L
	Chapter 2: Ecology	**6L**
	1. Elements of ecology: System, open system, closed system, definition of ecology, species, population, community, definition of ecosystem-components types and function.	1L
	2. Structure and function of the following ecosystem: Forest ecosystem, Grassland ecosystem, Desert ecosystem, Aquatic ecosystems, Mangrove ecosystem (special reference to Sundar ban); Food chain [definition and one example of each food chain], Food web.	2L
	3. Biogeochemical Cycle- definition, significance, flow chart of different cycles with only elementary reaction [Oxygen, carbon, Nitrogen, Phosphate, Sulphur].	1L
	4. Biodiversity- types, importance, Endemic species, Biodiversity Hot-spot, Threats to biodiversity, Conservation of biodiversity.	2L
	Chapter 3: Air pollution and control	**7L**
	1. Atmospheric Composition: Troposphere, Stratosphere, Mesosphere, Thermosphere, Tropopause and Mesopause	1L
	2. Energy balance: Conductive and Convective heat transfer, radiation heat transfer, simple global temperature model [Earth as a black body, earth as albedo], Problems.	1L
	3. Green house effects: Definition, impact of greenhouse gases on the global climate and consequently on sea water level, agriculture and marine food. Global warming and its consequence, Control of Global warming. Earth's heat budget.	1L
	4. Lapse rate: Ambient lapse rate Adiabatic lapse rate, atmospheric stability, temperature inversion (radiation inversion). Atmospheric dispersion: Maximum mixing depth, ventilation coefficient, effective stack height, smokestack plumes and Gaussian plume model.	1L

	5. Definition of pollutants and contaminants, Primary and secondary	1L
	pollutants: emission standard, criteria pollutant. Sources and effect of different air pollutants- Suspended particulate matter, oxides of carbon, oxides of nitrogen, oxides of sulphur, particulate, PAN.	
	6. Smog, Photochemical smog and London smog. Depletion Ozone layer: CFC, destruction of ozone layer by CFC, impact of other green house gases, effect of ozone modification.	1L
2	7. Standards and control measures: Industrial, commercial and residential air quality standard, control measure (ESP. Cyclone separator, bag house, catalytic converter, scrubber (ventury), Statement with brief reference).	1L
	Chapter 4: Water Pollution and Control	**8L**
	1. Hydrosphere, Hydrological cycle and Natural water.	1L
	2. Pollutants of water, their origin and effects: Oxygen demanding wastes, pathogens, nutrients, Salts, thermal application, heavy metals, pesticides, volatile organic compounds.	2L
	3. River/Lake/ground water pollution: River: DO, 5 day BOD test, Seeded BOD test, BOD reaction rate constants, Effect of oxygen demanding wastes on river[deoxygenation, reaeration], COD, Oil, Greases, pH.	1L
	4. Lake: Eutrophication [Definition, source and effect]. Ground water: Aquifers, hydraulic gradient, ground water flow (Definition only)	1L
	5. Standard and control: Waste water standard [BOD, COD, Oil, Grease], Water Treatment system [coagulation and flocculation, sedimentation and filtration, disinfection, hardness and alkalinity, softening] Waste water treatment system, primary and secondary treatments [Trickling filters, rotating biological contractor, Activated sludge, sludge treatment, oxidation ponds] tertiary treatment definition.	2L
	6. Water pollution due to the toxic elements and their biochemical effects: Lead, Mercury, Cadmium, and Arsenic	1L
3	**Chapter 5: Land Pollution**	**3L**
	1. Lithosphere; Internal structure of earth, rock and soil	1L
	2. Solid Waste: Municipal, industrial, commercial, agricultural, domestic, pathological and hazardous solid wastes; Recovery and disposal method- Open dumping, Land filling, incineration, composting, recycling. Solid waste management and control (hazardous and biomedical waste).	2L
	Chapter 5: Noise Pollution	**2L**
	1. Definition of noise, effect of noise pollution, noise classification [Transport noise, occupational noise, neighbourhood noise]	1L
	2. Definition of noise frequency, noise pressure, noise intensity, noise threshold limit value, equivalent noise level, $L10$ (18 hr Index) , $n\ Ld$, Noise pollution control.	1L
	Chapter 6: Environmental Management	**2L**
	1. Environmental impact assessment, Environmental Audit, Environmental laws and protection act of India, Different international environmental treaty/ agreement/ protocol.	2L
Total Number Of Hours = 34L		

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: **Basic Environmental Engineering** Subject Code: **CH-301**
Year: **2nd Year** Semester: **Third**

Faculty In-Charge HOD, ME Dept.

Assignment:

Module-1.
1. Write short notes for the following:
(a) Flood (b) Landslides (b) Earthquake (c) Acid Rain

2. Suppose an anemometer at a height of 40 m above ground measure wind velocity =5.5 m/s. Estimate the wind speed at an elevation of 500 m in rough terrain if atmosphere is unstable (i.e., k = 0.2).

Module-2.
1. A BOD test is run using 50 ml of wastewater mixed with 100 ml of pure water. The initial DO of the mixture is 6 mg/l and after 5 days it becomes 2 mg/l. After a long time, the DO remains fixed at 1 mg/l.
(i)What is the 5 days BOD (BOD_5)?
(ii)What is the ultimate BOD (BOD_u)?
(iii)What is the remaining BOD after 5 days?
(iv)What is the reaction rate constant measured at 20°C?
(v)What would be the reaction rate if measured at 35°C?

2. Draw the flow diagram for the following (a) Surface water treatment (b) Waste water Treatment.

3. Draw the Oxygen sag curve.

Module-3.
1. a) If two machines produces sounds of 80 dB and 120 dB simultaneously, what will be the total sound level.
b) Calculate the intensity of 100 dB sounds.

2. Write a report on the environmental problems related to an abandoned airport. Mention various measures by which it can be used again for other purposes.

UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

Lecture-wise Plan

Subject Name: Solid Mechanics **Subject Code-CE 301**
Year: 2nd Year **Semester: Third**

Module Number	Topics	Number of Lectures
1.	**Review of Basic Concepts of Stress and Strain:**	**8 L**
	Normal stress, Shear stress, Bearing stress,Normal	**2L**
	strain, Shearing strain; Hooke's law; Poisson's ratio;	**2L**
	Stress-strain diagram of ductile andbrittle materials;	**2L**
	Elastic limit; Ultimate stress; Yielding; Modulus of	**2L**
	elasticity; Factor of safety.	
2.	**Beam Statics**: Support reactions, concepts of	**8 L**
	redundancy, axial force, shear force and bending	**2L**
	moment diagrams for concentrated, uniformly	**2L**
	distributed, linearly varying load, concentrated	**2L**
	moments in simply supported beams, cantilever and	**2L**
	overhanging beams.	
3.	**Symmetric Beam Bending**: Basic kinematic	**9L**
	assumption, moment of inertia, elastic flexure	**2L**
	formulae and its application, Bending and shear	**2L**
	stress for regular sections, shear centre. Deflection of	**1L**
	statically determinate beams: Fundamental concepts:	**1L**
	Elastic curve, moment Curvature relationship,	**1L**
	governing differential equation, boundary conditions: Direct integration solution.	**2L**
4.	**Analysis of determinate plane trusses**: Concepts of	**9L**
	redundancy, Analysis by method of joints,method of	**3L**
	sections. Two Dimensional Stress Problems:	**3L**
	Principal stresses, maximum shear stresses, Mohr's	**2L**
	circle of stresses, construction of Mohr's circle.	**1L**
5.	**Introduction to thin cylindrical & spherical shells**:	**10L**
	Hoop stress and meridional - stress andvolumetric	**2L**
	changes.Torsion: Pure torsion, torsion of circular	**1L**
	solid shaft and hollow shafts, torsional equation,	**1L**
	torsional rigidity, closed coil helical; springs	**2L**
	Columns: Fundamentals, criteria for stability in	**1L**
	equilibrium, column buckling theory, Euler's load	**1L**
	for columns with different end conditions, limitations	**1L**
	of Euler's theory – problems, eccentric load and secant formulae.	**1L**
Total Number Of Hours = 44		

Faculty In-Charge HOD, CE Dept.

Assignment:
Module 1:

1. Explain the Analysis of bars of varying cross section in details mathematically. Derive the formula to obtain the total change in the length of the bar.

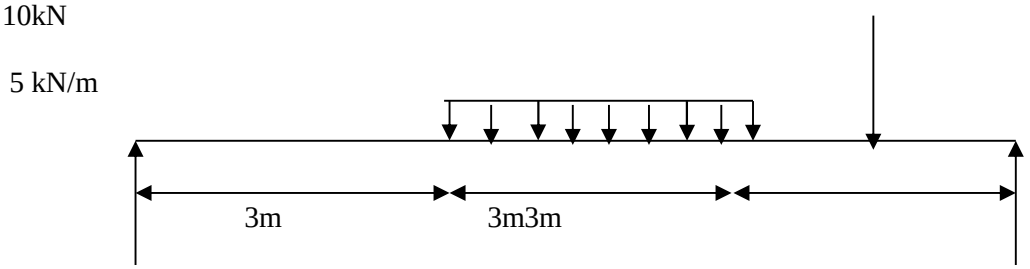
2. An Axial pull of 35,000N is acting on a bar consisting of three lengths as per the details given below. Young's Modulus =$2.1*10^5$ N/mm^2. Diameter of section 01=2 cm, Length of section 01=20cm, Diameter of section 02=3cm, Length of section 02=25cm, Diameter of section 03=5cm and Length of section 03=22cm. Determine the following:
a. Stresses in each section.
b. Total Extension of the bar.

3. a.)Explain the concept of Thermal Stress in details. Derive the mathematical expression for Thermal Stress with reference to Coefficient of thermal expansion.
b.)A rod is 2m long at a temperature of 10^oC. Find the expansion of the rod , when the temperature is raised to 80^oC. If this expansion is prevented, find the stress induced in the materials of the rod. Take E=$1.0*10^5MN/m^2$ and Coefficient of Thermal Expansion=0.000012 per degree centigrade.

Module 2 & 3:

1.a.)Explain the Theory of Simple Bending along with diagrams and special reference to the variation of the stresses in the layers with respect to the Neutral Axis layer.
b.) State the assumptions made in the Theory Of Simple Bending.

2.Draw SFD & BMD for the beam as shown in the figure. The concentrated load is action at the centre of the end span of the beam.



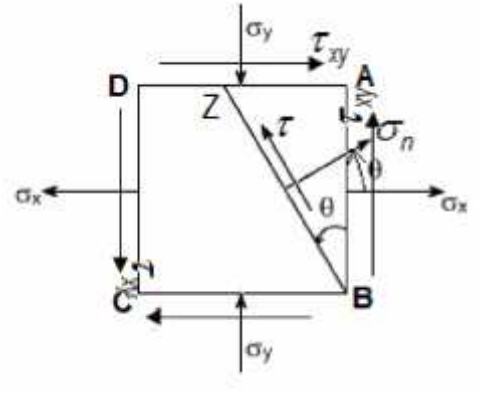
3. Derive the Expression for Bending Stress for the following parameters:

a.) Strain variation along the depth of the beam.
b.) Stress variation along the depth of the beam.
c.) Neutral Axis
d.) Moment Of Resistance and subsequently the Bending Equation.

Module 4:
1. Derive an equation for σ_n , τ

UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

Lecture-wise Plan



Module 5:

1. a.) Derive the maximum torque transmitted by a Circular Solid Shaft with reference to the total Turning Moment.

b.) A solid shaft of 150mm diameter is used to transmit torque. Find the maximum torque transmitted by the shaft if the maximum shear stress induced to the shaft is 45 N/mm^2.

c.)The shearing stress of a solid shaft is not to exceed $40N/mm^2$ when the Torque transmitted is 20,000N-m. Determine the minimum diameter of the shaft.

2. a.) Determine the total turning moment transmitted by a hollow circular shaft.

b.) Derive the power transmitted by shafts.

c.) In a hollow circular shaft of outer and inner diameter of 20cm and 10cm respectively, the shear stress is not to exceed $40N/mm^2$. Find the maximum torque which the shaft can safely transmit.

3. Explain the failure of a column with reference to the following points:

a.) Failure Of A Short Column.

b.) Failure Of A Long Column.

c.) Assumptions Made In The Euler's Column Theory.

d.) End Conditions For Long Columns.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: **Surveying** Subject Code: **CE302**

Year: **2nd Year** Semester: **Third**

Module Number	Topics	Number of Lectures
1	**Chapter 1: Introduction**	**2L**
	1. Introduction of surveying including definition, application and objectives.	1L
	2. Principle of surveying and classification.	1L
	Chapter 2: Chain surveying	**6L**
	1. Chain and its types,, Cross staff, Chaining for obtaining the outline of structures.	1L
	2. Plotting chain survey and Computation of areas.	1L
	3. Methods for overcoming obstacles, Conventional symbols.	1L
	4. Reconnaissance and site Location, Optical square	1L
	5. Locating ground features by offsets – Field book	1L
	6. Errors in chain surveying and their elimination: Problems	1L
	Chapter 3: Compass surveying	**7L**
	1. Bearings, Chain and compass surveying of an area,	1L
	2. Booking and plotting, Adjustments of traverse,	1L
	3. Problems in adjustments of traverse	1L
	4. Local attraction and its adjustments	1L
	5. Problems in local attraction	1L
	6. Errors in compass surveying and precautions: Problems.	1L
	7. Details of prismatic compass	1L
2	**Chapter 4: Plane table surveying**	**3L**
	1. Equipment, Orientation,	1L
	2. Methods of Plane Tabling,	1L
	3. Three Point Problems.	1L
	Chapter 5: Levelling	**8L**
	1. Introduction, Basic definitions,	1L
	2. Temporary adjustment of Levels, Sensitiveness of bubble tube;	1L
	3. Methods of leveling – Differential, Profile & fly Leveling,	1L
	4. Detail of dumpy Level, Automatic levels	1L
	5. Effect of curvature and refraction,,	1L
	6. Plotting longitudinal sections and Cross sections;	1L
	7. Measurement of area and volume	1L
	8. Problems in leveling	1L
	Chapter 5: Contouring	**4L**

	1. Introduction including definition and applications, Contour Interval.	1L
	2. Characteristics of Contour.	1L
	3. Methods of Locating Contours.	1L
	4. Topographic Map, Interpolation of Contours.	1L
3	**Chapter 6: The odolite surveying**	**3L**
	1. Introduction, Components of a Transit Theodolite.	1L
	2. Measurement of horizontal and vertical Angles, Co-ordinates and traverse Table.	1L
	3. Problems on theodolite traversing.	1L
	Chapter 7: Tacheometry	**3L**
	1. Definition, Details of stadia System,	1L
	2. Determination of horizontal and vertical 11 distance with Tacheometer- Staff held vertically and normal to the line of sight	1L
	3. Determination of horizontal and vertical 11 distance with Tacheometer- Staff held vertically and normal to the line of sight	1L
4	**Chapter 7: Simple and transition curves**	**3L**
	1. Definition, Degree of Curve, classification of horizontal curves, Elements of Simple Curve,	1L
	2. Problems on simple curve, Transition Curves.	1L
	3. Setting out by Linear method and Rankine's tangential method	1L
	Chapter 8: Total station	**1L**
	1. Introduction to Total Station with Field applications	1L
Total Number Of Hours = 40L		

Faculty In-Charge HOD, CE Dept.

Assignment:

Module-1

1. Calculate the back bearing of the following lines.

   a. N 12^0 24' W b. $145^0$15'

2. To determine the width river, a chain line PQR was laid across it, the points Q & R being on two sides of a river. From point S, 60 m from Q on line QS which was at right angles to PQ, the bearings of points R and P and were found to be 280^o and 190^0 respectively. If the distance PQ was 32 m, determine the distance QR and draw the sketch.

Module-2

1. The following offsets were taken at 15 m intervals from a survey line to an irregular boundary line:

3.50, 4.30, 6.75, 5.25, 7.50, 8.80, 7.90, 6.40, 4.40, 3.25 m

Calculate the area enclosed between the survey line, the irregular boundary line, and the first and the last offsets by Simpson's rule

2. The following consecutive readings were taken at the time of a levelling operation at intervals of 20 m with the help of dumpy level.
2.375, 1.730, 0.615, 3.450, 2.835, 2.070, 1.835, 0.985, 0.435, 1.630, 2.255 and 3.630 m.
The instrument was shifted after the 4th and 8th readings. The first reading was taken on a BM of RL 110.2 m. Find the RLs of all the points.

Module-3:

1. Prepare the Gale's table from the following data recorded during Theodolite traversing. (15)

Instrument Station	Interior angles	Line	Length (m)	WCB
A	73°31'0"	-	-	-
B	107°42'0"	AB	66.6	30°30'
C	187°8'0"	BC	135.7	102°47'36"
D	77°30'0"	CD	66.3	95°39'12"
E	94°7'0"	DE	76.6	198°8'48"

Module-4:

1. Describe the Rankine's method of setting a simple circular curve.

2. Two straight lines T_1I and T_2I intersect at chainage (375+12), the angle of deflection being 110^o. Calculate the chainage of the tangent points of a right-handed circular curve of 400 m radius, if 20 m chain was used.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: **Building Material & Construction & Material** Subject Code: **CE303**
Year: **2nd Year** Semester: **Third**

Module No.	Topics	Planned Lectures(H)
1.	**BRICKS & LIME**	**6 H**
	BRICKS	
	1. Classification, Characteristics of good bricks, Ingredients of good brick earth	1 H
	2. Harmful substance in brick Earth, Different forms of bricks,	1 H
	3. Testing of bricks as per BIS. Defects of bricks	1H
	LIME	
	1. Impurities in limestone, Classification, Slaking and hydration,	2H
	2. Hardening, Testing, Storage, Handling	1 H
2.	**AGGREGATE**	**3H**
	1. Classification, Characteristics,	1H
	2. Deleterious substances, Soundness, Alkali – aggregates reaction,	1H
	3. Fine aggregates, Coarse aggregates, Testing of aggregates	1 H
3.	**CEMENT & CONCRETE**	**6 H**
	CEMENT	
	1. OPC: Composition, PPC,	1H
	2. Slag cement, Hydration, setting time	1H
	CONCRETE	
	Types, ingredients, W/C ratio,	2 H
	Workability, Different grades in cement concrete,	1 H
	Tests on cement concrete	1 H
4.	**Mortars**	**3 H**
	1. Classification, Uses, Characteristics of good mortar,	1H
	2. Ingredients. Cement mortar, Lime mortar	1H
	3. Lime cement mortar, special mortars	1 H
5.	**WOOD AND WOOD PRODUCT**	**4 H**
	1. **Classification of Timber, Structure, Characteristics of good timber,**	1H
	2. Seasoning of timber, Defects in Timber,.	1H

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: **Building Material & Construction & Material** Subject Code: **CE-303**
Year: **2nd Year** Semester: **3rd**

	Preservation of Timber Testing of Timber	1 H
	Applications of wood and wood products	1 H
6.	Miscellaneous Materials: Gypsum	**3H**
	1. Classification, Plaster of Paris, Gypsum wall Plasters,	1H
	2. Gypsum Plaster Boards, Adhesives,	1H
	3. Heat and sound insulating materials, Geo-synthetics	1H
7.	**Foundations:**	**3 H**
	1. Function of Foundations, Essential requirement of good foundation,	2H
	2. Different types of shallow and deep Foundations	1H
8.	**Stairs**	**4H**
	1. Technical Terms, Requirements of good stair,	1H
	2. Dimension of steps	1H
	Classification, Geometric design of a dog legged stair case	2 H
9.	**Plastering and Pointing**	**3 H**
	Plastering with cement mortar, Defects in plastering,	1 H
	pointing, white washing, colour washing, Distempering,	2 H
10.	**Brick Masonary**	**3 H**
	1. Rules for bonding: stretcher bond, header bond.	2 H
	2. English and Flemish bonds for one, one and a half brick thick wall.	1 H
11.	**Roofs:**	**3H**
	Types, Pitched roofs and their sketches,	1 H
	Lean – to roof, King Post – Truss, Queen post truss	1 H
	and Simple steel Truss , Roof Covering materials: AC sheets GI sheet	1 H
TOTAL HOUR REQUIRED=44		

Faculty In-Charge HOD, CE
Dept.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: **Building Material & Construction & Material** Subject Code: **CE-303**
Year: **2nd Year** Semester: **3rd**

Assignment :

Module : 1

1. Write about composition of brick and manufacturing of brick.

2. What are the types of brick considering all aspect ?

3. Discuss about testing of brick?

4. Discuss about the importance of lime in manufacturing of concrete.

5. What are the types of lime?

6. Discuss about the manufacturing of lime.

7. Difference between Hydrated lime and Slaked lime.

Module :2

1. Write about characteristics of aggregate and importance of it.

2. Discuss about Deleterious substances, Soundness, Alkali – aggregates reaction.

Module :3

1. What are the composition of cement? Discuss in details.

2. Discuss about manufacturing of cement?

3. Discuss in brief about all the types of cement.

4. Define Slag of cement & hydration of cement.

5. Discuss about all the types of concrete with its uses.

6. Define Workability, Different grades in cement concrete, Creep of concrete, W/C ratio.

7. Discuss about Tests on cement concrete

Module :4

1. Discuss about different types of mortar mentioning their character and uses.

Module :5

1. Write short notes on:(i) seasoning of timber (ii) Decay of timber

(III) Defects of timber.

Module :6

1.Heat and sound insulating materials, Geo-synthetics

2. What are Properties and uses of Tar, Bitumen and Asphalt.

Module :7

1. What is foundation? Discuss in detail about the classification of foundation.

Module :8

1. Draw a R.C. Stair cases with sketches, Elevation and Cross section.

2. Design principles and design of a dog-legged stair case.

Module :9

1. Define Plastering with cement and lime mortar

2. What is White-washing, color washing and distempering

Module :10

1. Define brick masonry.

2. What are the types of brick masonry? Differentiate English & Flemish bond?

Module :11

1. Describe the Types, Pitched roofs and their sketches.

2. Draw a King Post – Truss, Queen post truss

3. Difference between A.C sheet & G.I sheet

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Data structure & Algorithm Lab
Course Code: CS(CE)391
L-T-P scheme: 0-0-3 **Course Credit: 2**

Objectives:

1. Develop problem solving ability using Programming.
2. Develop ability to design and analyze algorithms.
3. Introduce students to data abstraction and fundamental data structures.
4. Develop ability to design and evaluate Abstract Data Types and data structures.
5. Apply data structure concepts to various examples and real life applications

Learning Outcomes:

The course will use hands on practice and applying the knowledge gained in theory course to different day to day real world applications..Upon the completion of data structure and algorithm practical course, the student will be able to:

- **Understand** and implement different type of data structure techniques
- **Analyze** the hashing method.
- **Implement** different type of sorting searching techniques.

Course Contents:

Exercises that must be done in this course are listed below:

Exercise No.1: Implementation of array operations
Exercise No. 2: Implementation of linked lists: inserting, deleting a linked list.
Exercise No. 3: Stacks and Queues: adding, deleting elements
Exercise No. 4: Evaluation Problem: Evaluation of infix to postfix expressions on stack.
Exercise No. 5: Circular Queue: Adding & deleting elements
Exercise No. 6: Implementation of stacks using linked lists, Polynomial addition, Polynomial multiplication
Exercise No. 7: Sparse Matrices: Multiplication, addition.
Exercise No. 8: Recursive and Non-recursive traversal of Trees
Exercise No. 9: Threaded binary tree traversal. AVL tree implementation
Exercise No. 10: Application of sorting and searching algorithms

Text Book:

1. Yashavant Kanetkar, Abduln A.P.J. Kalam," Data Structure Through C",2nd edition, BPB Publications
2. Seymour Lipschutz,"Data Structures**",**Revised First edition,McGraw Hill Education.

Recommended Systems/Software Requirements:

1. Intel based desktop PC with minimum of 166 MHZ or faster processor with at least 64 MB RAM and 100 MB free disk space.
2. Turbo C or TC3 complier in Windows XP or Linux Operating System.

Exercise No.1: Implementation of array operations

Description:

An array is a collection of similar data elements. These data elements have the same data type.The elements of the array are stored in consecutive memory locations and are referenced by an `index`(also known as the subscript). The subscript is an ordinal number which is used to identify an element of the array.There are a number of operations that can be performed on arrays. These operations include:

KA Traversing an array
2) Inserting an element in an array
MA Searching an element in an array
NA Deleting an element from an array

OASorting an array in ascending or descending order

Aim: Write a program to insert a number at a given location in an array.

Algorithm:

The algorithm `INSERT` will be declared as `INSERT(A,N,POS,VAL)`. The arguments are

Step1: `A`, the array in which the element has to be inserted
Step2: `N`, the number of elements in the array
Step3: `POS`, the position at which the element has to be inserted
Step4: `VAL`, the value that has to be inserted

Program:

```
#include <stdio.h>
#include <conio.h>
int main()
{
 int i, n, num, pos, arr[10];
 clrscr();
 printf("\n Enter the number of elements in the array : ");
 scanf("%d", &n);
 for(i=0;i<n;i++)
 {
  printf("\n arr[%d] = ", i);
  scanf("%d", &arr[i]);
 }
 printf("\n Enter the number to be inserted : ");
 scanf("%d", &num);
 printf("\n Enter the position at which the number has to be added : scanf("%d", &pos);
 for(i=n–1;i>=pos;i—)
  arr[i+1] = arr[i];
 arr[pos] = num;
 n = n+1;
 printf("\n The array after insertion of %d is : ", num);
 for(i=0;i<n;i++)
 printf("\n arr[%d] = %d", i, arr[i]);
 getch();
 return 0;
}
```

Input:

```
 Enter the number of elements in the array : 5
 arr[0] = 1
 arr[1] = 2
 arr[2] = 3
 arr[3] = 4
 arr[4] = 5
 Enter the number to be inserted : 0
 Enter the position at which the number has to be added : 3
```

Output:

```
 The array after insertion of 0 is :
 arr[0] = 1
 arr[1] = 2
 arr[2] = 3
 arr[3] = 0
 arr[4] = 4
 arr[5] = 5
```

Aim:Write a program to delete a number from a given location in an array.

Algorithm:

The algorithm `DELETE` will be declared as `DELETE(A,N,POS)`. The arguments are:

Step2: N, the number of elements in the array
Step3: POS, the position from which the element has to be deleted

Program

```
#include <stdio.h>
#include <conio.h>
int main()
{
 int i, n, pos, arr[10];
 clrscr();
 printf("\n Enter the number of elements in the array : ");
 scanf("%d", &n);
 for(i=0;i<n;i++)
 {
  printf("\n arr[%d] = ", i);
  scanf("%d", &arr[i]);
 }
 printf("\nEnter the position from which the number has to be deleted : ");
 scanf("%d", &pos);
 for(i=pos; i<n–1;i++)
  arr[i] = arr[i+1];
 n—;
 printf("\n The array after deletion is : ");
 for(i=0;i<n;i++)
  printf("\n arr[%d] = %d", i, arr[i]);
 getch();
 return 0;
}
```

Input:

Enter the number of elements in the array : 5
arr[0] = 1
arr[1] = 2
arr[2] = 3
arr[3] = 4
arr[4] = 5
Enter the position from which the number has to be deleted : 3

Output:

The array after deletion is :
arr[0] = 1
arr[1] = 2
arr[2] = 3
arr[3] = 5

Lab assignment:

1) Merging two arrays
2) Sorting an array in ascending or descending order

Exercise No. 2: Implementation of linked lists: inserting, deleting a linked list.

Description:

A singly linked list is the simplest type of linked list in which every node contains some data anda pointer to the next node of the same data type. By saying that the node contains a pointer to the next node, we mean that the node stores the address of the next node in sequence.
A new node is added into an already existing linked list like
Case 1: The new node is inserted at the beginning.
Case 2: The new node is inserted at the end.

Case 4: The new node is inserted before a given node.
Before we describe the algorithms to perform insertions in all these four cases, let us first discuss an important term called OVERFLOW. Overflow is a condition that occurs when AVAIL = NULL or no free memory cell is present in the system. When this condition occurs, the program must give an appropriate message.
A node is deleted from an already existing linked list like
Case 1: The first node is deleted.
Case 2: The last node is deleted.
Case 3: The node after a given node is deleted.
Before we describe the algorithms in all these three cases, let us first discuss an important termCalled UNDERFLOW. Underflow is a condition that occurs when we try to delete a node from a linked list that is empty. This happens when START = NULL or when there are no more nodes to delete.
Note that when we delete a node from a linked list, we actually have to free the memory occupied by that node. The memory is returned to the free pool so that it can be used to store other programs and data. Whatever be the case of deletion, we always change the AVAIL pointer so that it points to the address that has been recently vacated.

Algorithm:

Insertion(A) Inserting a Node Before a Given Node in a Linked List

Step 1: IF AVAIL=NULL
Write OVERFLOWGo to Step 12
[END OF IF]
NEW_NODE
Step 2: SET = AVAIL
Step 3: SET AVAIL=AVAIL □□□NEXT
Step 4: SET NEW_NODE ->DATA=VAL
Step 5: SET PTR=START
Step 6: SET PREPTR=PTR
Step 7: Repeat Steps8and9while PTR DATA != NUM
Step 8: SET PREPTR=PTR
Step 9: SET PTR=PTR->NEXT
[END OF LOOP]
Step 10:PREPTR->NEXT = NEW_NODE
Step 11: SET NEW_NODE-> NEXT=PTR
Step 12: EXIT

Insertion(B) Inserting a Node After a Given Node in a Linked List
Step 1: IF AVAIL=NULL
Write OVERFLOW Go to Step 12
[END OF IF]
Step 2: SET = AVAIL->NEW_NODE
Step 3: SET AVAIL=AVAIL->NEXT
Step 4: SET DATA=VAL->NEW_NODE
Step 5: SET PTR=START
Step 6: SET PREPTR=PTR
Step 7: Repeat Steps8and9while PREPTR->DATA!= NUM
Step 8: SET PREPTR=PTR
Step 9: SET PTR=PTR->NEXT
[END OF LOOP]
Step 10: PREPTR->NEXT =NEW_NODE
Step 11: SET NEW_NODE->NEXT=PTR
Step 12: EXIT

Deletion
Step 1: IF START=NULL
Write UNDERFLOW
Go to Step 10
[END OF IF]
Step 2: SET PTR=START

Step 5: SET PREPTR=PTR
Step 6: SET PTR=PTR->NEXT
[END OF LOOP]
Step 7: SET TEMP=PTR
Step 8: SET PREPTR->NEXT=PTR->NEXT
Step 9: FREE TEMP
Step 10:EXIT

Aim:Write a program to create a linked list and perform insertions and deletions Write functions to sort and finally delete the entire list at once.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <malloc.h>
struct node
{
 int data;
 struct node *next;
};
struct node *start = NULL;
struct node *create_ll(struct node *);
struct node *display(struct node *);
struct node *insert_beg(struct node *);
struct node *insert_end(struct node *);
struct node *insert_before(struct node *);
struct node *insert_after(struct node *);
struct node *delete_beg(struct node *);
struct node *delete_end(struct node *);
struct node *delete_node(struct node *);
struct node *delete_after(struct node *);
struct node *delete_list(struct node *);
struct node *sort_list(struct node *);
int main(int argc, char *argv[]) {
 int option;
 do
 {
  printf("\n\n *****MAIN MENU *****");
  printf("\n 1:  Create a list");
  printf("\n 2:  Display the list");
  printf("\n 3:  Add a node at the beginning");
  printf("\n 4:  Add a node at the end");
  printf("\n 5:  Add a node before a given node");
  printf("\n 6:  Add a node after a given node");
  printf("\n 7:  Delete a node from the beginning");
printf("\n 8:  Delete a node from the end");
printf("\n 9:  Delete a given node");
printf("\n 10: Delete a node after a given node");
printf("\n 11: Delete the entire list");
printf("\n 12: Sort the list");
printf("\n 13: EXIT");
printf("\n\n Enter your option : ");
  scanf("%d", &option);
  switch(option)
  {
case 1: start = create_ll(start);
 printf("\n LINKED LIST CREATED");
```

```
case 2: start = display(start);
   break;
case 3: start = insert_beg(start);
   break;
case 4: start = insert_end(start);
   break;
case 5: start = insert_before(start);
   break;
case 6: start = insert_after(start);
   break;
case 7: start = delete_beg(start);
   break;
case 8: start = delete_end(start);
   break;
case 9: start = delete_node(start);
   break;
case 10: start = delete_after(start);
   break;
case 11: start = delete_list(start);
 printf("\n LINKED LIST DELETED");
   break;
case 12: start = sort_list(start);
   break;
  }
}while(option !=13);
return 0;
struct node *create_ll(struct node *start)
struct node *new_node, *ptr;
printf("\n Enter -1 to end");
printf("\n Enter the data : ");
scanf("%d", &num);
while(num!=-1)
new_node = (struct node*)malloc(sizeof(struct node));
new_node -> data=num;
  if(start==NULL)
  {
 new_node -> next = NULL;
   start =
new_node;
  }
  else
  {
   ptr=start;
   while(ptr->next!=NULL)
   ptr=ptr->next;
   ptr->next =
new_node;
   new_node->next=NULL;
  }
  printf("\n Enter the data : ");
  scanf("%d", &num);
 }
 return start;
}
struct node *display(struct node *start)
{
```

```
 ptr = start;
 while(ptr != NULL)
 {
  printf("\t %d", ptr -> data);
  ptr = ptr -> next;
 }
 return start;
}
struct node *insert_beg(struct node *start)
{
 struct node *new_node;
 int num;
 printf("\n Enter the data : ");
 scanf("%d", &num);
 new_node = (struct node *)malloc(sizeof(struct node));
 new_node -> data = num;
 new_node -> next = start;
 start = new_node;
 return start;
}
struct node *insert_end(struct node *start)
{
 struct node *ptr, *new_node;
 int num;
 printf("\n Enter the data : ");
 scanf("%d", &num);
 new_node = (struct node *)malloc(sizeof(struct node));
 new_node -> data = num;
 new_node -> next = NULL;
 ptr = start;
 while(ptr -> next != NULL)
 ptr = ptr -> next;
 ptr -> next = new_node;
 return start;
}
struct node *insert_before(struct node *start)
{
 struct node *new_node, *ptr, *preptr;
 int num, val;
 printf("\n Enter the data : ");
 scanf("%d", &num);
 printf("\n Enter the value before which the data has to be inserted : ");
 scanf("%d", &val);
 new_node = (struct node *)malloc(sizeof(struct node));
 new_node -> data = num;
 ptr = start;
 while(ptr -> data != val)
 {
  preptr = ptr;
  ptr = ptr -> next;
 }
 preptr -> next = new_node;
 new_node -> next = ptr;
 return start;
}
struct node *insert_after(struct node *start)
```

```
 struct node *new_node, *ptr, *preptr;
 int num, val;
 printf("\n Enter the data : ");
 scanf("%d", &num);
 printf("\n Enter the value after which the data has to be inserted : ");
scanf("%d", &val);
new_node = (struct node *)malloc(sizeof(struct node));
new_node -> data = num;
ptr = start;
preptr = ptr;
while(preptr -> data != val)
 {
 preptr = ptr;
 ptr = ptr -> next;
 }
preptr -> next=new_node;
new_node -> next = ptr;
return start;
struct node *delete_beg(struct node *start)
struct node *ptr;
ptr = start;
start = start -> next;
 free(ptr);
return start;
struct node *delete_end(struct node *start)
struct node *ptr, *preptr;
ptr = start;
while(ptr -> next != NULL)
 {
 preptr = ptr;
 ptr = ptr -> next;
 }
preptr -> next = NULL;
 free(ptr);
return start;
struct node *delete_node(struct node *start)
struct node *ptr, *preptr;
int val;
printf("\n Enter the value of the node which has to be deleted : ");
scanf("%d", &val);
ptr = start;
if(ptr -> data == val)
 {
start = delete_beg(start);
return start;
}
 else
 {
while(ptr -> data != val)
  {
   preptr = ptr;
   ptr = ptr -> next;
  }
  preptr -> next = ptr -> next;
  free(ptr);
  return start;
```

```
}
struct node *delete_after(struct node *start)
{
 struct node *ptr, *preptr;
 int val;
 printf("\n Enter the value after which the node has to deleted : ");
 scanf("%d", &val);
 ptr = start;
 preptr = ptr;
 while(preptr -> data != val)
 {
  preptr = ptr;
  ptr = ptr -> next;
 }
 preptr -> next=ptr -> next;
 free(ptr);
 return start;
}
struct node *delete_list(struct node *start)
{
            struct
            node     *ptr;
 if(start!=NULL){
  ptr=start;
  while(ptr != NULL)
  {
   printf("\n %d is to be deleted next", ptr -> data);
   start =
delete_beg(ptr);
   ptr =
start;
  }
 }

 return start;
}
struct node *sort_list(struct node *start)
{
 struct node *ptr1, *ptr2;
 int temp;
 ptr1 = start;
 while(ptr1 -> next != NULL)
 {
  ptr2 = ptr1 -> next;
  while(ptr2 != NULL)
  {
   if(ptr1 -> data > ptr2 -> data)
   {
            temp = ptr1 -> data;
            ptr1 -> data = ptr2 -> data;
            ptr2 -> data = temp;
   }
   ptr2 = ptr2 -> next;
  }
  ptr1 = ptr1 -> next;
 }
```

```
}
```

Input:

```
3
4
5
```

Output:

```
*****MAIN MENU *****
1: Create a list
2: Display the list
3: Add a node at the beginning
4: Add the node at the end
5: Add the node before a given node
6: Add the node after a given node
7: Delete a node from the beginning
8: Delete a node from the end
9: Delete a given node
10: Delete a node after a given node
11: Delete the entire list
12: Sort the list
13: Exit

Enter your option : 1
Enter the data :3
Enter your option : 2
3
Enter your option : 3
Enter the data : 4
Enter your option : 6
Add after given node:4
Enter the data : 5
Enter your option : 2
4       5       3
Enter your option : 10
Delete after a given node:5
Enter your option : 2
4       5
```

Lab Assignment:

1) WAP to implement circular linked list.
2) WAP to insert and delete an element in a doubly linked list(all cases).

Exercise No. 3: Stacks and Queues: adding, deleting elements

Description:

A stack is a linear data structure which uses the same principle, i.e., the elements in a stack are added and removed only from one end, which is called the`TOP`. Hence, a stack is called a LIFO (Last-In First-Out) datastructure, as the element that was inserted last is the first one to be taken out.

A stack supports three basic operations: `push`, `pop`, and `peek`. The `push` operation adds an element to the top of the stack and the `pop` operation removes the element from the top of the stack. The `peek` operation returns the value of the topmost element of the stack.

Aim: Write a program to perform Push, Pop, and Peek operations on a stack.

Algorithm:

Insertion:

```
Step 1: IF TOP=MAX-1
                PRINT OVERFLOW
        Goto Step 4
```

Step 2: SET TOP=TOP+1
Step 3: SET STACK[TOP]=VALUE
Step 4: END

Deletion:
Step 1: IF TOP=NULL
PRINT UNDERFLOW
Goto Step 4
[END OF IF]
Step 2: SET VAL=STACK[TOP]
Step 3: SET TOP=TOP-1
Step 4: END

Peek:
Step 1: IF TOP=NULL
PRINT STACK IS EMPTY
Goto Step 3
Step 2: RETURN STACK[TOP]
Step 3: END

Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define MAX 3 // Altering this value changes size of stack created
int st[MAX], top=-1;
void push(int st[], int val);
int pop(int st[]);
int peek(int st[]);
void display(int st[]);
int main(int argc, char *argv[]) {
 int val, option;
 do
 {
  printf("\n *****MAIN MENU*****");
  printf("\n 1. PUSH");
  printf("\n 2. POP");
  printf("\n 3. PEEK");
  printf("\n 4. DISPLAY");
  printf("\n 5. EXIT");
  printf("\n Enter your option: ");
  scanf("%d", &option);
  switch(option)
  {
  case 1:
   printf("\n Enter the number to be pushed on stack: ");
   scanf("%d", &val);
   push(st, val);
   break;
  case 2:
   val = pop(st);
   if(val != -1)
   printf("\n The value deleted from stack is: %d", val);
   break;
  case 3:
```

```
   if(val != -1)
   printf("\n The value stored at top of stack is: %d", val);
   break;
  case 4:
   display(st);
   break;
  }
 }while(option != 5);
 return 0;
}
void push(int st[], int val)
{
 if(top == MAX-1)
 {
  printf("\n STACK OVERFLOW");
 }
 else
 {
  top++;
  st[top] = val;
 }
}
int pop(int st[])
{
 int val;
 if(top == -1)
 {
  printf("\n STACK UNDERFLOW");
  return -1;
 }
 else
 {
  val = st[top];
  top--;
  return val;
 }
}
void display(int st[])
{
 int i;
 if(top == -1)
 printf("\n STACK IS EMPTY");
 else
 {
  for(i=top;i>=0;i--)
  printf("\n %d",st[i]);
  printf("\n"); // Added for formatting purposes
 }
}
int peek(int st[])
{
 if(top == -1)
 {
  printf("\n STACK IS EMPTY");
  return -1;
 }
```

```
 return (st[top]);
}
```

Output

```
*****MAIN MENU*****
1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
Enter your option : 1
Enter the number to be pushed on stack : 500
Enter your option : 1
Enter the number to be pushed on stack : 700
Enter your option : 4
700     500
Enter your option : 3
Enter your option : 4
700
Enter your option : 2
Enter your option : 4
500
```

Description:

A queue is a FIFO (First-In, First-Out) data structure in which the element that is inserted first is the first one to be taken out. The elements in a queue are added at one end called the REAR and removed from the other end called the FRONT. Queues can be implemented by using either arrays or linked lists.

Aim: Write a program to perform Insertion, Deletion, and Peek operations on a queue.

Algorithm:

Insertion:

```
Step 1: IF REAR=MAX-1
                Write OVERFLOW
        Goto step 4
        [END OF IF]
Step 2: IF FRONT=-1 and REAR=-1
                SET FRONT=REAR =ELSE
                SET REAR=REAR+1
        [END OF IF]
Step 3: SET QUEUE[REAR]=NUM
Step 4: EXIT
```

Deletion:

```
Step 1: IF FRONT=-1OR FRONT>REAR
                Write UNDERFLOW
        ELSE
                SET VAL=QUEUE[FRONT]
                SET FRONT=FRONT+1
        [END OF IF]
Step 2: EXIT
```

Program:

```
#include <stdio.h>
#include <conio.h>
#define MAX 10 // Changing this value will change length of array
int queue[MaX];
int front = -1, rear = -1;
void insert(void);
int delete_element(void);
int peek(void);
void display(void);
int main()
```

```
 do
 {
  printf("\n\n ***** MAIN MENU *****");
  printf("\n 1. Insert an element");
  printf("\n 2. Delete an element");
  printf("\n 3. Peek");
  printf("\n 4. Display the queue");
  printf("\n 5. EXIT");
  printf("\n Enter your option : ");
  scanf("%d", &option);
  switch(option)
  {
  case 1:
   insert();
   break;
  case 2:
   val = delete_element();
   if (val != -1)
   printf("\n The number deleted is : %d", val);
   break;
  case 3:
   val = peek();
   if (val != -1)
   printf("\n The first value in queue is : %d", val);
   break;
  case 4:
   display();
   break;
  }
 }while(option != 5);
 getch();
 return 0;
}
void insert()
{
 int num;
 printf("\n Enter the number to be inserted in the queue : ");
 scanf("%d", &num);
 if(rear == MAX-1)
 printf("\n OVERFLOW");
 else if(front == -1 && rear == -1)
 front = rear = 0;
 else
 rear++;
 queue[rear] = num;
}
int delete_element()
{
 int val;
if(front == -1 || front>rear)
 {
 printf("\n UNDERFLOW");
 return -1;
 }
 else
 {
 val = queue[front];
 front++;
 if(front > rear)
 front = rear = -1;
 return val;
```

```
if(front==-1 || front>rear)
 {
 printf("\n QUEUE IS EMPTY");
 return -1;
 }
 else
 {
 return queue[front];
 }
void display()
int i;
printf("\n");
if(front == -1 || front > rear)
printf("\n QUEUE IS EMPTY");
 else
 {
 for(i = front;i <= rear;i++)
 printf("\t %d", queue[i]);
 }
```

Output:

```
***** MAIN MENU *****"
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit
Enter your option : 1
Enter the number to be inserted in the queue : 50
```

Exercise No. 4: Evaluation Problem: Evaluation of infix to postfix expressions on stack.

Description:

Infix, postfix, and prefix notations are three different but equivalent notations of writing algebraic expressions. For example, if an expression is written as `A+B` in infix notation, the same expression can be written as `AB+` in postfix notation. The order of evaluation of a postfix expression is always from left to right. Even brackets cannot alter the order of evaluation. The expression `(A+B)*C` can be written as: `[AB+]*C =>AB+C*` in the postfix notation.

Aim:Write a program to convert a given infix expression into its postfix Equivalent, Implement the stack using an array.

Algorithm:

Step 1: Add)to the end of the infix expression
Step 2: Push(onto the stack
Step 3: Repeat until each character in the infix notation is scanned
IF a(is encountered, push it on the stack
IF an operand (whetheradigit oracharacter) is encountered, add it to thepostfix expression.
IF a)is encountered, then
    a. Repeatedly pop from stack and add it to the postfix expression until a
(is encountered.
    b. Discard the (.That is, remove the(from stack and do notadd it to the postfix expression
IF an operator is encountered, then
    a. Repeatedly pop from stack and add each operator (popped from the stack) to thepostfix expression
which has the same precedence orahigher precedence than)
    b. Push the operator to the stack
[END OF IF]
Step 4: Repeatedly pop from the stack and add it to the postfix expression until the stack is empty
Step 5: EXIT

Program:

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
```

```
char stack[MAX];
int top=1;
char pop();  /*declaration of  pop function*/
void push(char item); /*declaration of  push function*/
int prcd(char symbol)  /*checking the precedence*/
{
   switch(symbol) /*assigning values for symbols*/
   {
      case '+':
      case '-': return 2;
      break;
      case '*':
      case '/': return 4;
      break;
      case '^':return 6;
      break;
      case '(':
      case ')':
      case '#':return 1;
     break;
  }
}
int(isoperator(char symbol))  /*assigning  operators*/
{
   switch(symbol)
   {
       case '+':
       case '*':
   case '-':
      case '/':
      case '^':
      case '(':
      case ')':return 1;
      break;
      default:return 0;
   }
 }
/*converting infix to postfix*/
void convertip(char infix[],char postfix[])
{
 int i,symbol,j=0;
 stack[++top]='#';
 for(i=0;i<strlen(infix);i++)
 {
  symbol=infix[i];
  if(isoperator(symbol)==0)
  {
     postfix[j]=symbol;
      j++;
  }
  else
  {
     if(symbol=='(')
     push(symbol);   /*function call  for pushing elements into the stack*/
     else if(symbol==')')
     {
```

```
      {
         postfix[j]=pop();
         j++;
      }
      pop();     /*function call  for popping elements into the stack*/
   }
  else
  {
      if(prcd(symbol)>prcd(stack[top]))
      push(symbol);
      else
      {
        while(prcd(symbol)<=prcd(stack[top]))
        {
          postfix[j]=pop();
           j++;
        }
       push(symbol);
      }/*end of else loop*/
    }/*end of else loop*/
   } /*end of else loop*/
  }/*end of for loop*/
 While (stack[top]!='#')
 {
    postfix[j]=pop();
     j++;
  }
  postfix[j]='\0'; /*null terminate string*/
}
/*main program*/
void main()
 {
     char infix[20],postfix[20];
     printf("enter the valid infix string \n");
     gets(infix);
     convertip(infix,postfix);    /*function call   for converting infix to postfix */
     printf("the corresponding postfix string is:\n");
    puts(postfix);
  }
/*push operation*/
void push(char item)
{
   top++;
   stack[top]=item;
 }
/*pop operation*/
char pop()
{
   char a;
   a=stack[top];
   top--;
   return a;
 }
```

Input:

A+B*C

Output:

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Exercise No. 5: Circular Queue: Adding & deleting elements

Description:

In the circular queue, the first index comes right after the last index.The circular queue will be full only when `FRONT=0` and `REAR=Max-1`. A circular queue is implemented in the same manner as a linear queue is implemented.

Aim: Write a program to implement a circular queue using array.

Algorithm:

Insertion:

```
Step 1: IF FRONT = and Rear=MAX-1
                Write OVERFLOW
                Goto step 4
        [End OF IF]

Step 2:
IF FRONT=-1 and REAR=-1
        SET FRONT=REAR =0
ELSE IF REAR=MAX-1and FRONT !=0
                SET REAR =0
   ELSE
                SET REAR=REAR+1
[END OF IF]
Step 3: SET QUEUE[REAR]=VAL
Step 4: EXIT
Deletion:
Step 1: IF FRONT=-1
                Write UNDERFLOW
                Goto Step 4
        [END of IF]
Step 2: SET VAL=QUEUE[FRONT]
Step 3: IF FRONT=REAR
                SET FRONT=REAR=-1
        ELSE
                IF FRONT=MAX -1
                        SET FRONT =0
                ELSE
                        SET FRONT=FRONT+1
                [END of IF]
        [END OF IF]
Step 4: EXIT
```

Program:

```
#include <stdio.h>
#include <conio.h>
#define MAX 10
int queue[MAX];
int front=–1, rear=–1;
void insert(void);
int delete_element(void);
int peek(void);
void display(void);
int main()
{
 int option, val;
 clrscr();
 do
 {
  printf("\n ***** MAIN MENU *****");
  printf("\n 1. Insert an element");
  printf("\n 2. Delete an element");
  printf("\n 3. Peek");
```

```
 printf("\n Enter your option : ");
 scanf("%d", &option);
 switch(option)
 {
  case 1:
   insert();
   break;
  case 2:
   val = delete_element();
   if(val!=–1)
   printf("\n The number deleted is : %d", val);
   break;
  case 3:
   val = peek();
   if(val!=–1)
   printf("\n The first value in queue is : %d",     break;
  case 4:
   display();
   break;
 }
}while(option!=5);
getch();
return 0;
}
void insert()
{
 int num;
 printf("\n Enter the number to be inserted in the queue : ");
 scanf("%d", &num);
 if(front==0 && rear==MAX–1)
  printf("\n OVERFLOW");
 else if(front==–1 && rear==–1)
 {
  front=rear=0;
  queue[rear]=num;
 }
 else if(rear==MAX–1 && front!=0)
 {
  rear=0;
  queue[rear]=num;
 }
 else
 {
  rear++;
  queue[rear]=num;
 }
}
int delete_element()
{
 int val;
 if(front==–1 && rear==–1)
  {
  printf("\n UNDERFLOW");
  return –1;
  }
 val = queue[front];
 if(front==rear)
  front=rear=–1;
 else
 {
  if(front==MAX–1)
```

```
  front++;
 }
 return val;
}
int peek()
{
 if(front==–1 && rear==–1)
 {
  printf("\n QUEUE IS EMPTY");
  return –1;
}
 else
 {
  return queue[front];
 }
}
void display()
{
 int i;
 printf("\n");
 if (front ==–1 && rear= =–1)
  printf ("\n QUEUE IS EMPTY");
 else
 {
  if(front<rear)
  {
   for(i=front;i<=rear;i++)
    printf("\t %d", queue[i]);
  }
  else
  {
   for(i=front;i<MAX;i++)
    printf("\t %d", queue[i]);
   for(i=0;i<=rear;i++)
    printf("\t %d", queue[i]);
  }
 }
}
```

Output

```
***** MAIN MENU *****
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. EXIT
Enter your option : 1
Enter the number to be inserted in the queue : 25
Enter your option : 2
The number deleted is : 25
Enter your option : 3
QUEUE IS EMPTY
Enter your option : 5
```

Exercise No. 6: Implementation of Polynomial addition, Polynomial multiplicationusing linked lists.

Description:

A polynomial is represented in the memory using a linked list. Consider a polynomial $6x^3+9x^2+7x+1$. Every individual term in a polynomial consists of two parts, a coefficientand a power. Here, 6, 9, 7, and 1 are the coefficients of the terms that have 3, 2, 1, and 0 as theirpowers respectively.

Every term of a polynomial can be represented as a node of the linked list

Aim:Write a program to add two polynomials.

Program:

```
#include <stdio.h>
typedef struct pnode
{
float coef;
int exp;
struct pnode *next;
}p;
p *getnode();
void main()
{
p *p1,*p2,*p3;

p *getpoly(),*add(p*,p*);

void display(p*);
clrscr();
printf("\n enter first polynomial");
p1=getpoly();
printf("\n enter second polynomial");
p2=getpoly();
printf("\nthe first polynomial is");
display(p1);
printf("\nthe second polynomial is");
display(p2);
p3=add(p1,p2);
printf("\naddition of two polynomial is :\n");
display(p3);

}
p *getpoly()
{
p *temp,*New,*last;
int flag,exp;
char ans;
float coef;
temp=NULL;
flag=1;
printf("\nenter the polynomial in descending order of exponent");
do
{
printf("\nenter the coef & exponent of a term");
scanf("%f%d",&coef,&exp);
New=getnode();
if(New==NULL)
printf("\nmemory cannot be allocated");
New->coef=coef;
New->exp=exp;
if(flag==1)
{
temp=New;
last=temp;
flag=0;
}
else
{
last->next=New;
last=New;
}
```

```
ans=getch();
}
while(ans=='y');
return(temp);
}
p *getnode()
{
p *temp;
temp=(p*) malloc (sizeof(p));
temp->next=NULL;
return(temp);
}
void display(p*head)
{
p*temp;
temp=head;
if(temp==NULL)
printf("\npolynomial empty");
while(temp->next!=NULL)
{
printf("%0.1fx^%d+",temp->coef,temp->exp);
temp=temp->next;
}
printf("\n%0.1fx^%d",temp->coef,temp->exp);
getch();
}
p*add(p*first,p*second)
{
p *p1,*p2,*temp,*dummy;
char ch;
float coef;
p *append(int,float,p*);
p1=first;
p2=second;
temp=(p*)malloc(sizeof(p));
if(temp==NULL)
printf("\nmemory cannot be allocated");
dummy=temp;
while(p1!=NULL&&p2!=NULL)
{
if(p1->exp==p2->exp)
{
coef=p1->coef+p2->coef;
temp=append(p1->exp,coef,temp);
p1=p1->next;
p2=p2->next;
}
else
if(p1->expexp)
{
coef=p2->coef;
temp=append(p2->exp,coef,temp);
p2=p2->next;
}
else
if(p1->exp>p2->exp)
{
coef=p1->coef;
temp=append(p1->exp,coef,temp);
p1=p1->next;
}
```

```
{
temp=append(p1->exp,p1->coef,temp);
p1=p1->next;
}
while(p2!=NULL)
{
temp=append(p2->exp,p2->coef,temp);
p2=p2->next;
}
temp->next=NULL;
temp=dummy->next;
free(dummy);
return(temp);
}
p*append(int Exp,float Coef,p*temp)
{
p*New,*dum;
New=(p*)malloc(sizeof(p));
if(New==NULL)
printf("\ncannot be allocated");
New->exp=Exp;
New->coef=Coef;
New->next=NULL;
dum=temp;
dum->next=New;
dum=New;
return(dum);
}
```

Input:

A^2+2A+2

A^3+3A+3

Output:

A^3+A^2+5A+5

Lab Assignment:

1) Write a program to multiply two polynomials.

Exercise No. 7: Sparse Matrices: Multiplication, addition.

Description:

Sparse matrix is a matrix that has large number of elements with a zero value. In order to efficiently utilize the memory, specialized algorithms and data structures that take advantage of the sparse structure should be used. If we apply the operations using standard matrix structures and algorithms to sparse matrices, then the execution will slow down and the matrix will consume large amount of memory. Sparse data can be easily compressed, which in turn can significantly reduce memory usage.

Aim: Write a program to multiply sparse matrices.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
#define MAX1 3
#define MAX2 3
#define MAXSIZE 20
#define TRUE 1
#define FALSE 2
struct sparse
{
int *sp ;
int row ;
```

```
} ;
void initsparse ( struct sparse * ) ;
void create_array ( struct sparse * ) ;
int count ( struct sparse ) ;
void display ( struct sparse ) ;
void create_tuple ( struct sparse*, struct sparse ) ;
void display_tuple ( struct sparse ) ;
void prodmat ( struct sparse *, struct sparse, struct sparse ) ;
void searchina ( int *sp, int ii, int*p, int*flag ) ;
void searchinb ( int *sp, int jj, int colofa, int*p, int*flag ) ;
void display_result ( struct sparse ) ;
void delsparse ( struct sparse * ) ;
void main( )
{
struct sparse s[5] ;
int i ;
clrscr( ) ;
for ( i = 0 ; i<= 3 ; i++ )
initsparse ( &s[i] ) ;
create_array ( &s[0] ) ;
create_tuple ( &s[1], s[0] ) ;
display_tuple ( s[1] ) ;
create_array ( &s[2] ) ;
create_tuple ( &s[3], s[2] ) ;
display_tuple ( s[3] ) ;
prodmat ( &s[4], s[1], s[3] ) ;
printf ( "\nResult of multiplication of two matrices: " ) ;
display_result ( s[4] ) ;
for ( i = 0 ; i<= 3 ; i++ )
delsparse ( &s[i] ) ;
getch( ) ;
}
/* initialises elements of structure */
void initsparse ( struct sparse *p )
{
p -> sp = NULL ;
p -> result = NULL ;
}
/* dynamically creates the matrix */
void create_array ( struct sparse *p )
{
int n, i ;
/* allocate memory */
p -> sp = ( int * ) malloc ( MAX1 * MAX2 * sizeof ( int ) ) ;
/* add elements to the array */
for ( i = 0 ; i< MAX1 * MAX2 ; i++ )
{
printf ( "Enter element no. %d: ", i ) ;
scanf ( "%d", &n ) ;
* ( p -> sp + i ) = n ;
}
}
/* displays the contents of the matrix */
```

```
{
int i ;
/* traverses the entire matrix */
for ( i = 0 ; i< MAX1 * MAX2 ; i++ )
{
/* positions the cursor to the new line for every new row */
if ( i % 3 == 0 )
printf ( "\n" ) ;
printf ( "%d\t", * ( s.sp + i ) ) ;
}
}
/* counts the number of non-zero elements */
int count ( struct sparse s )
{
int cnt = 0, i ;
for ( i = 0 ; i< MAX1 * MAX2 ; i++ )
{
if ( * ( s.sp + i ) != 0 )
cnt++ ;
}
return cnt ;
}
/* creates an array that stores information about non-zero elements */
void create_tuple ( struct sparse *p, struct sparse s )
{
int r = 0 , c = -1, l = -1, i ;
/* get the total number of non-zero elements */
p -> row = count ( s ) + 1 ;
/* allocate memory */
p -> sp = ( int * ) malloc ( p -> row * 3 * sizeof ( int ) ) ;
/* store information about total no. of rows, cols, and non-zero values */
* ( p -> sp + 0 ) = MAX1 ;
* ( p -> sp + 1 ) = MAX2 ;
* ( p -> sp + 2 ) = p -> row - 1 ;
l = 2 ;
/* scan the array and store info. about non-zero values in the 3-tuple */
for ( i = 0 ; i< MAX1 * MAX2 ; i++ )
{
c++ ;
/* sets the row and column values */
if ( ( ( i % 3 ) == 0 ) && ( i != 0 ) )
{
r++ ;
c = 0 ;
}
/* checks for non-zero element, row, column and non-zero value is assigned to the matrix */
if ( * ( s.sp + i ) != 0 )
{
l++ ;
* ( p -> sp + l ) = r ;
l++ ;
* ( p -> sp + l ) = c ;
l++ ;
```

```
}
}
}
/* displays the contents of the matrix */
void display_tuple ( struct sparse s )
{
int i, j ;
/* traverses the entire matrix */
printf ( "\nElements in a 3-tuple: " ) ;
j = ( * ( s.sp + 2 ) * 3 ) + 3 ;
for ( i = 0 ; i< j ; i++ )
{
/* positions the cursor to the new line for every new row */
if ( i % 3 == 0 )
printf ( "\n" ) ;
printf ( "%d\t", * ( s.sp + i ) ) ;
}
printf ( "\n" ) ;
}
/* performs multiplication of sparse matrices */
void prodmat ( struct sparse *p, struct sparse a, struct sparse b )
{
int sum, k, position, posi, flaga, flagb, i , j ;
k = 1 ;
p -> result = ( int * ) malloc ( MAXSIZE * 3 * sizeof ( int ) ) ;
for ( i = 0 ; i< * ( a.sp + 0 * 3 + 0 ) ; i++ )
{
for ( j = 0 ; j< * ( b.sp + 0 * 3 + 1 ) ; j++ )
{
/* search if an element present at ith row */
searchina ( a.sp, i, &position, &flaga ) ;
if ( flaga == TRUE )
{
sum = 0 ;
/* run loop till there are element at ith row in first 3-tuple */
while ( * ( a.sp + position * 3 + 0 ) == i )
{
/* search if an element present at ith col. in second 3-tuple */
searchinb ( b.sp, j, * ( a.sp + position * 3 + 1 ), &posi, &flagb ) ;
/* if found then multiply */
if ( flagb == TRUE )
sum = sum + * ( a.sp + position * 3 + 2 ) * * ( b.sp + posi * 3 + 2 ) ;
position = position + 1 ;
}
/* add result */
if ( sum != 0 )
{
* ( p -> result + k * 3 + 0 ) = i ;
* ( p -> result + k * 3 + 1 ) = j ;
* ( p -> result + k * 3 + 2 ) = sum ;
k = k + 1 ;
}
}
```

```
}
/* add total no. of rows, cols and non-zero values */
* ( p -> result + 0 * 3 + 0 ) = * ( a.sp + 0 * 3 + 0 ) ;
* ( p -> result + 0 * 3 + 1 ) = * ( b.sp + 0 * 3 + 1 ) ;
* ( p -> result + 0 * 3 + 2 ) = k - 1 ;
}
/* searches if an element present at iith row */
void searchina ( int *sp, int ii, int *p, int *flag )
{
int j ;
*flag = FALSE ;
for ( j = 1 ; j<= * ( sp + 0 * 3 + 2 ) ; j++ )
{
if ( * ( sp + j * 3 + 0 ) == ii )
{
*p = j ;
*flag = TRUE ;
return ;
}
}
}
/* searches if an element where col. of first 3-tuple is equal to row of second 3-tuple */
void searchinb ( int *sp, int jj, int colofa, int *p, int *flag )
{
int j ;
*flag = FALSE ;
for ( j = 1 ; j<= * ( sp + 0 * 3 + 2 ) ; j++ )
{
if ( * ( sp + j * 3 + 1 ) == jj && * ( sp + j * 3 + 0 ) == colofa )
{
*p = j ;
*flag = TRUE ;
return ;
}
}
}
/* displays the contents of the matrix */
void display_result ( struct sparse s )
{
int i ;
/* traverses the entire matrix */
for ( i = 0 ; i< ( * ( s.result + 0 + 2 ) + 1 ) * 3 ; i++ )
{
/* positions the cursor to the new line for every new row */
if ( i % 3 == 0 )
printf ( "\n" ) ;
printf ( "%d\t", * ( s.result + i ) ) ;
}
}
/* deallocates memory */
void delsparse ( struct sparse *s )
{
if ( s -> sp != NULL )
```

```
if ( s -> result != NULL )
free ( s -> result ) ;
}
```

Input:

First matrices

```
[0 2 3]
[4 0 0]
[0 0 5]
```

Second matrices

```
[0 0 7]
[0 8 0]
[0 9 6]
```

Output:

```
[0 43 18]
[0  0 28]
[0 45 30]
```

Lab assignment:

1) Write a program to add two sparse matrices.

Exercise No. 8: Recursive and Non-recursive traversal of Trees

Description:

A binary tree is a data structure that is defined as a collection of elements called nodes. In a binary tree, the topmost element is called the root node, and each node has 0, 1, or at the most 2 children. A node that has zero children is called a leaf node or a terminal node. Every node contains a data element, a left pointer which points to the left child, and a right pointer which points to the right child. The root element is pointed by a `'root'` pointer. If `root = NULL`, then it means the tree is empty.

Aim: Write a program to implement a binary tree using recursion.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
struct node
{
int data;
struct node *left,*right;
};
struct node *root;
void insert(int x)
{
  struct node *p,*previous,*current;
  p=(struct node *)malloc(sizeof(struct node));
  if(p==NULL)
  {
   printf("\n Out of memory");
  }
  p->data=x;
  p->left=NULL;
  p->right=NULL;
  if(root=NULL)
  {
   root=p;
   return;
  }
  previous=NULL;
  current=root;
  while(current!=NULL)
  {
     previous=current;
     if(p->data<current->data)
```

```
            current=current->right;
            }
     if(p->data<previous->data)
         previous->left=p;
     else
 previous->right=p;
}
void inorder(struct node *t)
{
  if (t!=NULL)
  {
  inorder(t->left);
  printf("\n %5d",t->data);
  inorder (t->right);
  }
}
void del(int x)
{
   int tright=0,tleft=0;
   struct node *ptr=root;
   struct node *parent=root;
   struct node *t1=root;
   struct node *temp=root;
   while(ptr!=NULL&& ptr->data!=x)
   {
      parent=ptr;
      if (x<ptr->data)
           ptr=ptr->left;
      else
          ptr=ptr->right;
   }
   if (ptr==NULL)
   {
     printf("\n Delete element not found");
     return ;
   }
   else if(t1->data==x && (t1->left ==NULL || t1->right==NULL))
         if(t1->left==NULL)
           t1=t1->right;
         else
           t1=t1->left;
   else if (ptr->left==NULL)
     if (x<parent->data)
       parent->left=ptr->right;
     else
       parent->right=ptr->right;
   else if (ptr->right==NULL)
     if (x<parent->data)
       parent->left=ptr->left;
     else
       parent->right=ptr->left;
   else
   {
   temp=ptr;
   parent=ptr;
   if((ptr->left)>=(ptr->right))
   {
      ptr=ptr->left;
      while(ptr->right!=NULL)
      {
          tright=1;
```

```
    }
    temp->data=ptr->data;
    if(tright)
        parent->right=ptr->left;
    else
        parent->left=ptr->left;
  }
 else
  {
   ptr=ptr->right;
   while (ptr->left!=NULL)
   {
        tleft=1;
        parent=ptr;
        ptr=ptr->left;
    }
    temp->data=ptr->data;
    if(tleft)
        parent->left=ptr->right;
    else
        parent->right=ptr->right;
  }
  free(ptr);
 }
}

void main()
{
int op,n,srchno;
root=(struct node *)malloc(sizeof(struct node));
root->data=30;
root->right=root->left=NULL;
clrscr();
do
{
   printf("\n 1.Insertion");
   printf("\n 2.Deletion");
   printf("\n 3.Inorder");
   printf("\n 4.Quit");
   printf("\n Enter your choice\n");
   scanf("%d",&op);

 switch (op)
  {
   case 1: printf("\n Enter the element to insert\n");
           scanf("%d",&n);
           insert(n);
           break;
   case 2: printf("\n Enter the element to be deleted\n");
           scanf("%d",&srchno);
           del(srchno);
           break;
   case 3: printf("\n The inorder elements are\n");
           inorder(root);
           getch();
           break;
   default: exit(0);
  }
 }while(op<4);
 getch();

}
```

1 2 3

Output:

Enter the element to insert1
Enter the element to insert2
Enter the element to insert3
The inorder elements are
2 1 3

Lab assignment:

1) Write a program to implement a binary tree without using recursion

Exercise No. 9: AVL tree implementation

Description:

An AVL tree is the same as that of a binary search tree but with a little difference.

In its structure, it stores an additional variable called theBalance Factor. Thus, every node has a balance factor associated with it. The balance factor of a node is calculated by subtracting the height of its right sub-tree from the height of its left sub-tree. A binary search tree in which every node has a balance factor of –1, 0, or 1 is said to be height balanced. A node with any other balance factor is considered to be unbalanced and requires rebalancing of the tree.

Balance factor = Height (left sub-tree) – Height (right sub-tree)

Aim: Write a program to implement AVL tree

Program:

```
#include <stdio.h>
typedef enum { FALSE ,TRUE } bool;
struct node
{
 int val;
 int balance;
 struct node *left_child;
 struct node *right_child;
};
struct node* search(struct node *ptr, int data)
{
 if(ptr!=NULL)
   if(data < ptr -> val)
   ptr = search(ptr -> left_child,data);
   else if( data > ptr -> val)
   ptr = search(ptr -> right_child, data);
 return(ptr);
}
struct node *insert (int data, struct node *ptr, int *ht_inc)
{
 struct node *aptr;
struct node *bptr;
if(ptr==NULL)
 {
 ptr = (struct node *) malloc(sizeof(struct node));
 ptr -> val = data;
 ptr -> left_child = NULL;
 ptr -> right_child = NULL;
 ptr -> balance = 0;
 *ht_inc = TRUE;
 return (ptr);
 }
if(data < ptr -> val)
 {
 ptr -> left_child = insert(data, ptr -> left_child, ht_inc);
 if(*ht_inc==TRUE)
  {
  switch(ptr -> balance)
```

```
case -1: /* Right heavy */


ptr -> balance = 0;
   *ht_inc = FALSE;
   break;


case 0: /* Balanced */


ptr -> balance = 1;


break;
   case 1: /* Left heavy */


aptr = ptr -> left_child;
    if(aptr -> balance == 1)
     {


printf("Left to Left Rotation\n");


ptr -> left_child= aptr -> right_child;


aptr -> right_child = ptr;


ptr -> balance = 0;


aptr -> balance=0;
     ptr = aptr;
     }
    else
     {
     printf("Left to right rotation\n");


bptr = aptr -> right_child;
     aptr -> right_child = bptr -> left_child;


bptr -> left_child = aptr;


ptr -> left_child = bptr -> right_child;


bptr -> right_child = ptr;


if(bptr -> balance == 1 )


pt
```

```
else


pt
r -> balance = 0;


if(bptr -> balance == -1)


aptr -> balance = 1;


else


aptr -> balance = 0;


bptr -> balance=0;


ptr = bptr;
     }
    *ht_inc = FALSE;
    }
  }
 }
if(data > ptr -> val)
{
  ptr -> right_child = insert(info, ptr -> right_child, ht_inc);
  if(*ht_inc==TRUE)
  {
   switch(ptr -> balance)
   {


case 1: /* Left heavy */
     ptr -> balance = 0;
     *ht_inc = FALSE;
     break;


case 0: /* Balanced */
     ptr -> balance = -1;
     break;
    case -1: /* Right heavy */

 aptr = ptr -> right_child;
  if(aptr -> balance == -1)
     {
    printf("Right to Right Rotation\n");
  ptr -> right_child= aptr -> left_child;
    aptr -> left_child = ptr;
      ptr -> balance = 0;
     aptr -> balance=0;
  ptr = aptr;
     }
  else
     {
```

```
  aptr -> left_child = bptr -> right_child;
bptr -> right_child = aptr;
ptr -> right_child = bptr -> left_child;
  bptr -> left_child = pptr;
if(bptr -> balance == -1)
ptr -> balance = 1;
      else
ptr -> balance = 0;
      if(bptr -> balance == 1)
  aptr -> balance = -1;
      else
  aptr -> balance = 0;
      bptr -> balance=0;
  ptr = bptr;
}/*End of else*/
*ht_inc = FALSE;
   }
  }
 }
 return(ptr);
}
void display(struct node *ptr, int level)
{
 int i;
 if ( ptr!=NULL )
 {
  display(ptr -> right_child, level+1);
  printf("\n");
  for (i = 0; i < level; i++)
   printf(" ");
  printf("%d", ptr -> val);
  display(ptr -> left_child, level+1);
 }
}
void inorder(struct node *ptr)
{
 if(ptr!=NULL)
 {
  inorder(ptr -> left_child);
  printf("%d ",ptr -> val);
  inorder(ptr -> right_child);
 }
}
main()
{
 bool ht_inc;
 int data ;
 int option;
 struct node *root = (struct node *)malloc(sizeof(struct node));
root = NULL;
while(1)
 {
 printf("1.Insert\n");
 printf("2.Display\n");
 printf("3.Quit\n");
 printf("Enter your option : ");
 scanf("%d",&option);
 switch(choice)
   {
   case 1:
   printf("Enter the value to be inserted : ");
```

```
    root = insert(data, root, &ht_inc);
 else
 printf("Duplicate value ignored\n");
  break;
 case 2:
   if(root==NULL)
    {
 printf("Tree is empty\n");
    continue;
    }
   printf("Tree is :\n");
   display(root, 1);
 printf("\n\n");
printf("Inorder Traversal is: ");
 inorder(root);
   printf("\n");
   break;
  case 3:
   exit(1);
 default:
 printf("Wrong option\n");
    }
  }
 }
```

Input:

```
6       11      2       4       3       5
```

Output:

```
2       3       5       4       6       11
```

Lab Assignment:

1) Write a program to implement AVL tree

Exercise No. 10: Application of sorting and searching algorithms

Description:

To search an element in an array is known as searching and to sort the element in an ascending and descending order is known as sorting.Two type of searching linear and binary. Mainly five type of sorting like bubble ,insertion ,selection, merge and quick sort.here we mainly focus on binary search and merge and quick sort.

Aim:Implement Binary search without using recursion

Program:

```
#include<stdio.h>

int main(){

   int a[10],i,n,m,c=0,l,u,mid;

   printf("Enter the size of an array: ");
   scanf("%d",&n);

   printf("Enter the elements in ascending order: ");
   for(i=0;i<n;i++){
       scanf("%d",&a[i]);
   }

   printf("Enter the number to be search: ");
   scanf("%d",&m);

   l=0,u=n-1;
   while(l<=u){
       mid=(l+u)/2;
```

```
            c=1;
            break;
        }
        else if(m<a[mid]){
            u=mid-1;
        }
        else
            l=mid+1;
    }
    if(c==0)
        printf("The number is not found.");
    else
        printf("The number is found.");

    return 0;
}
```

OUTPUT:

```
Enter the size of an array: 5
        Enter the element in ascending order: 2  4  8  9  12
        Enter the number to be  search: 3
        The number is not found.
```

Aim: Implement Merge Sort using Divide and Conquer approach

Program:

```
#include<stdio.h>
#include<conio.h>
void merge(int [],int ,int ,int );
void part(int [],int ,int );
int main()
{
 int arr[30];
 int i,size;
 printf("\n\t------- Merge sorting method -------\n\n");
 printf("Enter total no. of elements : ");
 scanf("%d",&size);
 for(i=0; i<size; i++)
 {
   printf("Enter %d element : ",i+1);
   scanf("%d",&arr[i]);
 }
 part(arr,0,size-1);
 printf("\n\t------- Merge sorted elements -------\n\n");
 for(i=0; i<size; i++)
 printf("%d ",arr[i]);
 getch();
 return 0;
}

void part(int arr[],int min,int max)
{
 int mid;
 if(min<max)
```

```
  mid=(min+max)/2;
  part(arr,min,mid);
  part(arr,mid+1,max);
  merge(arr,min,mid,max);
 }
}

void merge(int arr[],int min,int mid,int max)
{
  int tmp[30];
  int i,j,k,m;
  j=min;
  m=mid+1;
  for(i=min; j<=mid && m<=max ; i++)
  {
    if(arr[j]<=arr[m])
    {
       tmp[i]=arr[j];
       j++;
    }
    else
    {
       tmp[i]=arr[m];
       m++;
    }
  }
  if(j>mid)
  {
    for(k=m; k<=max; k++)
    {
       tmp[i]=arr[k];
       i++;
    }
  }
  else
  {
    for(k=j; k<=mid; k++)
    {
      tmp[i]=arr[k];
      i++;
    }
  }
  for(k=min; k<=max; k++)
    arr[k]=tmp[k];
}
```

Output:

```
Enter the no of elements:7
7 8 9 4 5 3 1
The unsorted list is: 7 8 9 4 5 3 1
```

1 3 4 5 7 8 9

Aim:Implement Quick Sort using Divide and Conquer approach
Program:

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#define MAX 6000

void quick(int x[],int lb,int ub);
int partition(int x[],int lb,int ub);

void main()
{
        int i,n,x[MAX];
        time_t start,end;
        clrscr();
        printf("Enter the number of elements: ");
        scanf("%d",&n);

        for(i=0;i<n;i++)
                x[i]=rand();

        printf("\nEntered array is \n");
        for(i=0;i<n;i++)
                printf("%d ",x[i]);

        start=time(NULL);
        quick(x,0,n-1);
        end=time(NULL);
        printf("Sorted array is as shown:\n");
        for(i=0;i<n;i++)
                printf("%d ",x[i]);
        printf("\nTIME for %d elements : %f", n, difftime(end,start));
        getch();

}

void quick(int x[],int lb,int ub)
{
        int j;
        if(lb<ub)
        {
                printf("\n");
                j=partition(x,lb,ub);
                quick(x,lb,j-1);
                quick(x,j+1,ub);
        }
}
```

```
{
        int a,down,up,temp;
        a=x[lb];
        up=ub;
        down=lb;
        while(down<up)
        {
                while(x[down]<=a&&down<ub)
                        down++;
                while(x[up]>a)
                        up--;
                if(down<up)
                {
                        temp=x[down];
                        x[down]=x[up];
                        x[up]=temp;
                }
        }
        x[lb]=x[up];
        x[up]=a;
        return up;
}
```

Output:

```
Enter the number of elements:5
    Entered array is
      41   18467    6334    26500    19169
    Sorted array is as shown
      41   6334     18467    19169    26500
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

Title of Course: Solid Mechanics Lab
Course Code: CE 391
L-T-P scheme: 0-0-3 **Course Credit: 2**

Objectives:
1. To understand the basic concepts of mechanics of materials.
2. To discover the fundamental concepts of stress and strain.
3. To learn the practical relationship between load applied and the response developed by the material.

Learning Outcomes: The students will be able to develop and understand the clear concepts of stress and strain. The various tension and compression test performed on the material gives them a clear differentiation between the nature of loads and the response involved. The impact testing gives them a clear understanding of the sudden applied loading under sudden impact loading and the response developed by the structures. The student will also have a concept of fatigue loading and failure of the specimens due to fatigue loading.

Course Contents:
Practicalsthat must be done in this course are listed below:

- Tension test on Structural Materials: Mild Steel and Tor steel (HYSD bars).
- Compression Test on Structural Materials: Timber, bricks and concrete cubes.
- Bending Test on Mild Steel.
- Torsion Test on Mild Steel Circular Bar.
- Hardness Tests on Ferrous and Non-Ferrous Metals: Brinnel and Rockwell Tests.
- Test on closely coiled helical spring.
- Impact Test: Izod and Charpy.
- Demonstration of Fatigue Test.

Text Book:
1. Fundamentals Of Strength Of Materials S Ramamrutham.

EXPERIMENT NO. – 01

AIM: - Study of Universal Testing Machine (U.T.M.)

OBJECT: - To Study the various component parts of the Universal TestingMachine (U.T.M.) & test procedures of various practical's to be performed.

APPARATUS: - Universal Testing Machine with all attachment i.e. shears testattachment, bending attachment, tension grips, compression test attachment etc.

THEORY: - The Universal Testing Machine consists of two units.1) Loading unit, 2) Control panel.

LOADING UNIT:-
It consists of main hydraulic cylinder with robust base inside. The piston whichmoves up and down. The chain driven by electric motor which is fitted on lefthand side. The screw column maintained in the base can be rotated using abovearrangement of chain.

Each column passes through the main nut which is fitted inthe lower cross head. The lower table connected to main piston through a ball &the ball seat is joined to ensure axial loading.

There is a connection betweenlower table and upper head assembly that moves up and down with main piston.The measurement of this assembly is carried out by number of bearings whichslides over the columns. The test specimen each fixed in the job is known as‘Jack Job’. To fix up the specimen tightly, the movement of jack job is achievedhelically by handle.

CONTROL PANEL:-

It consists of oil tank having a hydraulic oil level sight glass for checking theoil level. The pump is displacement type piston pump having free plungers thoseensure for continuation of high pressure. The pump is fixed to the tank frombottom.

The suction & delivery valve are fitted to the pump near tank Electricmotor driven the pump is mounted on four studs which is fitted on the right sideof the tank. There is an arrangement for loosing or tightening of the valve. Thefour valves on control panel control the oil stroke in the hydraulic system.

Theloading system works as described below.The return valve is close, oil delivered by the pump through the flowcontrol valves to the cylinder & the piston goes up. Pressure starts developing &either the specimen breaks or the load having maximum value is controlled withthe base dynameters consisting in a cylinder in which the piston reciprocates.

The switches have upper and lower push at the control panel for the downward& upward movement of the movable head. The on & off switch provided on thecontrol panel & the pilot lamp shows the transmission of main supply.

METHOD OF TESTING:-

Initial Adjustment: - before testing adjust the pendulum with respect tocapacity of the test i.e. 8 Tones; 10 Tones; 20 Tones; 40 Tones etc.For ex: - A specimen of 6 tones capacity gives more accurate result of 10 Tonescapacity range instead of 20 Tones capacity range.

These ranges of capacity areadjusted on the dial with the help of range selector knob. Engineering controlweights of the pendulum are adjusted correctly. The ink should be inserted inpen holder of recording paper around the drum & the testing process is starteddepending upon the types of test as mentioned below.

TENSION TEST:-

Select the proper job and complete upper and lower check adjustment.Apply some Greece to the tapered surface of specimen or groove. Then operatethe upper cross head grip operation handle & grip the upper end of test specimenfully in to the groove.

Keep the lower left valve in fully close position. Open theright valve & close it after lower table is slightly lifted. Adjust the lower points tozero with the help of adjusting knob. This is necessary to remove the dead weightof the lower table.

Then lock the jobs in this position by operating job workinghandle. Then open the left control valve. The printer on dial gauge at which the specimen breaks slightly return back & corresponding load is known as breakingload & maximum load is known as the ultimate load.

COMPRESSION TEST:-

Fix upper and lower pressure plates to the upper stationary head & lower table respectively. Place the specimen on the lower plate in order to grip.Then adjust zero by lifting the lower table. Then perform the test in the samemanner as described in tension test.

FLEXURAL OR BENDING TEST:-
Keep the bending table on the lower table in such a way that the central positionof the bending table is fixed in the central location value of the lower table. Thebending supports are adjusted to required distance.Stuffers at the back of the bending table at different positions. Then place thespecimen on bending table & apply the load by bending attachment at the upperstationary head. Then perform the test in the same manner as described in tensiontest.

BRINELL HARDNESS TEST:-
Place the specimen on the lower table & lift it up slightly. Adjust thezero fixed value at the bottom side of the lower cross head. Increase the loadslowly ultimate load value is obtained. Then release the load slowly with leftcontrol valve. Get the impression of a suitable value of five to ten milli meter onthe specimen & measure the diameter of the impression correctly by microscope& calculate Brinell hardness.

SHEAR TEST:-
Place the shear test attachment on the lower table, this attachment consists ofcutter. The specimen is inserted in roles of shear test attachment & lift the lowertable so that the zero is adjusted, then applies the load such that the specimenbreaks in two or three pieces. If the specimen breaks in two pieces then it will bein angle shear, & if it breaks in three pieces then it will be in double shear.

STUDY OF EXTENSOMETER:-
This instrument is an attachment to Universal / Tensile Testing Machines. Thismeasures the elongation of a test place on load for the set gauge length. Theleast count of measurement being 0.01 mm, and maximum elongationmeasurement up to 3 mm. This elongation measurement helps in finding out theproof stress at the required percentage elongation.

WORKING OF THE INSTRUMENT:-
The required gauge length (between 30to 120) is set by adjusting the upperknife edges .A scale is provided for this purpose. Hold the specimen in theupper and lower jaws of Tensile / Universal Testing Machine. Position theextensometer on the specimen, Position upper clamp to press upper knifeedges on the specimen.

The extensometer will be now fixed to the specimen byspring pressure. Set zero on both the dial gauges by zero adjusts screws .Startloading the specimen and take the reading of load on the machine at requiredelongation or the elongation at required load.

Force setter accuracies mean ofboth the dial gauge readings should be taken as elongation. It is veryimportant to note & follow the practice of removing the extensometer from thespecimen before the specimen breaks otherwise the instrument will be totallydamaged. As a safety, while testing the instrument may be kept hanging from afixed support by a slightly loose thread.

TECHNICAL DATA:-

Measuring Range: 0 – 3mm.
Least Count: 0. 01 mm.
Gauge Length adjustable from: 30 – 120 mm
Specimen Size: 1 to 20mm Round or Flats up to 20 x 20 mm.

A) Stress-strain graph of Mild Steel
B) Stress-strain graphs of different materials.

• Curve **A** shows a **brittle** material. This material is also strong because there is littlestrain for a high stress. The fracture of a brittle material is sudden and catastrophic,with little or no plastic deformation. Brittle materials crack under tension and thestress increases around the cracks. Cracks propagate less under compression.

• Curve **B** is a **strong** material which is not ductile. Steel wires stretch very little, andbreak suddenly. There can be a lot of elastic strain energy in a steel wire undertension and it will "whiplash" if it breaks. The ends are razor sharp and such afailure is very dangerous indeed.

• Curve **C** is a **ductile** material.

• Curve **D** is a **plastic** material. Notice a very large strain for a small stress. Thematerial will not go back to its original length.

EXPERIMENT NO. – 02

AIM: -To determine tensile test on a metal.

OBJECT: - To conduct a tensile test on a mild steel specimen and determine thefollowing:

I. Limit of proportionality
II. Elastic limit
III. Yield strength
IV. Ultimate strength
V. Young's modulus of elasticity
VI. Percentage elongation
VII. Percentage reduction in area

APPARATUS: -

I. Universal Testing Machine (UTM)
II. Mild steel specimens
III. Graph paper
IV. Scale
V. VernierCaliper

THEORY:-The tensile test is most applied one, of all mechanical tests. In thistest ends of test piece are fixed into grips connected to a straining device and to aload measuring device. If the applied load is small enough, the deformation ofany solid body is entirely elastic. An elastically deformed solid will return to itsoriginal form as soon as load is removed. However, if the load is too large, thematerial can be deformed permanently.

The initial part of the tension curvewhich is recoverable immediately after unloading is termed. As elastic and therest of the curve which represents the manner in which solid undergoes plasticdeformation is termed plastic. The stress below which the deformationsessentially entirely elastic is known as the yield strength of material.

In somematerial the onset of plastic deformation is denoted by a sudden drop in loadindicating both an upper and a lower yield point. However, some materials donot exhibit a sharp yield point. During plastic deformation, at larger extensionsstrain hardening cannot compensate for the decrease in section and thus the loadpasses through a maximum and then begins to decrease.

This stage the "ultimatestrength"' which is defined as the ratio of the load on the specimen to originalcross-sectional area, reaches a maximum value. Further loading will eventuallycause 'neck' formation and rupture.

PROCEDURE:-
1) Measure the original length and diameter of the specimen. Thelength may either be length of gauge section which is marked on thespecimen with a preset punch or the total length of the specimen.
2. Insert the specimen into grips of the test machine and attachstrain-measuring device to it.
3. Begin the load application and record load versus elongationdata.
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.

OBESERVATION: - A) Material:
A) Original dimensions
Length = ----------
Diameter = ---------
Area = ----------
B) Final Dimensions:
Length = ----------
Diameter = ----------
Area = ----------

EXPERIMENT NO-03

AIM: - Hardness Test of Mild Steel.

OBJECT: - To conduct hardness test on mild steel, carbon steel, brass and aluminium specimens.

APPARATUS: - Hardness tester, soft and hard mild steel specimens, brass, aluminium etc.

THEORY: - The hardness of a material is resistance to penetration under a localizedpressure or resistance to abrasion. Hardness tests provide an accurate, rapid andeconomical way of determining the resistance of materials to deformation. There arethree general types of hardness measurements depending upon the manner in whichthe test is conducted:

a. Scratch hardness measurement,
b. Rebound hardness measurement

c. Indention hardness measurement.

In scratch hardness method the material are rated on their ability to scratch oneanother and it is usually used by mineralogists only. In rebound hardnessmeasurement, a standard body is usually dropped on to the material surface and thehardness is measured in terms of the height of its rebound .The general means ofjudging the hardness is measuring the resistance of a material to indentation.

Theindenters usually a ball cone or pyramid of a material much harder than that beingused. Hardened steel, sintered tungsten carbide or diamond indenters are generallyused in indentation tests; a load is applied by pressing the indenter at right angles tothe surface being tested.

The hardness of the material depends on the resistance whichit exerts during a small amount of yielding or plastic. The resistance depends onfriction, elasticity, viscosity and the intensity and distribution of plastic strainproduced by a given tool during indentation

PROCEDURE:-
1. Place the specimen securely upon the anvil.
2. Elevate the specimen so that it come into contact with the penetrate and putthe specimen under a preliminary or minor load of 100+2N without shock.
3. Apply the major load 900N by loading lever.
4. Watch the pointer until it comes to rest.
5. Remove the major load.
6. Read the Rockwell hardness number or hardness scale.

EXPERIMENT No:-04

AIM: - Torsion test on mild steel rod.

OBJECT: -To conduct torsion test on mild steel or cast iron specimens tofind out modulus of rigidity

APPARATUS: -
1. A torsion testing machine.
2. Twist meter for measuring angles of twist
3. A steel rule and VernierCaliper or micrometer.

PROCEDURE:-
1. Select the driving dogs to suit the size of the specimen and clamp it inthe machine by adjusting the length of the specimen by means of asliding spindle.
2. Measure the diameter at about three places and take theaverage value.
3. Choose the appropriate range by capacity change lever
4. Set the maximum load convenience and clamp it by means of knurled screw pointer to zero.
5. Set the protector to zero.
6. Carry out straining by rotating the handweel in either direction.
7. Load the machine in suitable increments.
8. Then load out to failure as to cause equal increments of strain reading.
9. Plot a torque- twist (T- θ) graph.
10. Read off co-ordinates of a convenient point from the straight line portion of the torque twist (T- θ) graph and calculate the value of Cby using relation.

EXPERIMENT No: - 05

AIM: - To determined impact strength of steel**.**

OBJECT: -To determine the impact strength of steel by Izod impacttest.

APPARATUS: -
1.Impact testing machine
2. A steel specimen 75 mm X 10mm X 10mm

THEORY:-
An impact test signifies toughness of material that is ability of materialto absorb energy during plastic deformation. Static tension tests of unnotchedspecimens do not always reveal the susceptibility of a metal to brittle fracture.This important factor is determined by impact test. Toughness takes intoaccount both the strength and ductility of the material.

Several engineeringmaterials have to withstand impact or suddenly applied loads while in service.Impact strengths are generally lower as compared to strengths achieved underslowly applied loads. Of all types of impact tests, the notch bar tests are mostextensively used. Therefore, the impact test measures the energy necessary tofracture a standard notch bar by applying an impulse load. The test measuresthe notch toughness of material under shock loading. Values obtained fromthese tests are not of much utility to design problems directly and are highlyarbitrary.

Still it is important to note that it provides a good way of comparingtoughness of various materials or toughness of the same material underdifferent condition. This test can also be used to assess the ductile brittletransition temperature of the material occurring due to lowering of temperature.

PROCEDURE:-

(a) lzod test
1. With the striking hammer (pendulum) in safe test position, firmlyhold the steel specimen in impact testing machine's vice in such away that the notch face the hammer and is half inside and halfabove the top surface of the vice.

2. Bring the striking hammer to its top most striking position unless itis already there, and lock it at that position.

3. Bring indicator of the machine to zero, or follow the instructions ofthe operating manual supplied with the machine.

4. Release the hammer. It will fall due to gravity and break thespecimen through its momentum, the total energy is not absorbedby the specimen. Then it continues to swing. At its topmost heightafter breaking the specimen, the indicator stops moving, while thependulum falls back. Note the indicator at that topmost finalposition.
5. Again bring back the hammer to its idle position and back

OBESERVATION:-

Izod Test.
1. Impact value of - Mild Steel ----------N-m
2. Impact value of - Brass ----------N-m
3. Impact value of - Aluminum ----------N-m

RESULT:-
i. The energy absorbed for Mild Steel is found out to be Joules.
ii. The energy absorbed for Brass is found out to be Joules.
iii. . The energy absorbed for Aluminum is found out to be Joules.

PRECAUTION:-
1. Measure the dimensions of the specimen carefully.
2. Hold the specimen (lzod test) firmly.
3. Note down readings carefully.

EXPERIMENT No: - 06

AIM: -To determined impact strength of steel**.**

OBJECT: -To determine the impact strength of steel by (Charpy test)

APPARATUS: -
1. Impact testing machine
2. A steel specimen 10 mm x 10 mm X 55mm

THEORY:-An impact test signifies toughness of material that is ability of materialto absorb energy during plastic deformation. Static tension tests of unmatchedspecimens do not always reveal the susceptibility of a metal to brittle fracture. Thisimportant factor is determined by impact test.

Toughness takes into account both thestrength and ductility of the material. Several engineering materials have towithstand impact or suddenly applied loads while in service. Impact strengths aregenerally lower as compared to strengths achieved under slowly applied loads. Of alltypes of impact tests, the notch bar tests are most extensively used. Therefore, theimpact test measures the energy necessary to fracture a standard notch bar by applying an impulse load.

The test measures the notch toughness of material undershock loading. Values obtained from these tests are not of much utility to designproblems directly and are highly arbitrary. Still it is important to note that it providesa good way of comparing toughness of various materials or toughness of the samematerial under different condition. This test can also be used to assess theEngineering ductile brittle transition temperature of the material occurring due tolowering of temperature.

PROCEDURE :-

(a) Charpy Test

1. With the striking hammer (pendulum) in safe test position, firmly holdthe steel specimen in impact testing machines vice in such a way thatthe notch faces s the hammer and is half inside and half above the topsurface of the vice.

2. Bring the striking hammer to its top most striking position unless it isalready there, and lock it at that position.

3. Bring indicator of the machine to zero, or follow the instructions of theoperating manual supplied with the machine.

4. Release the hammer. It will fall due to gravity and break the specimenthrough its momentum, the total energy is not absorbed by thespecimen. Then it continues to swing. At its topmost height afterbreaking the specimen, the indicator stops moving, while thependulum falls back. Note the indicator at that topmost final position.

5. The specimen is placed on supports or anvil so that the blow of hammeris opposite to the notch.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Surveying Practice-I Lab
Course Code: CE 392
L-T-P scheme: 0-0-3 **Course Credit: 2**

Objectives:
1. To learn and understand the concepts and perform the surveying of construction with the help of different methods of surveying like chain surveying, compass surveying, plane table surveying and levelling.
2. To understand and differentiate between compass surveying, plane table surveying, levelling and contouring.
3. To operate the various instruments of surveying like surveyors compass, to measure the distance between any two inaccessible points, to efficiently operate the dumpy levels and understand the level difference between inaccessible points.

Learning Outcomes: The students will have a clear understanding of the basic concepts of surveying. They will develop a clear understanding of the various steps of surveying with different instruments. Surveying is the first step before any construction activity and the students will be able to develop the concepts of chain surveying, compass surveying, plane table surveying, levelling and contouring.

Course Contents:
Practicals that must be done in this course are listed below:

1. Chain surveying
Preparing index plans, Location sketches, Ranging, Preparation of map, Heights of objects using chain and ranging rods, Getting outline of the structures by enclosing them in triangles/quadrilaterals, Distance between inaccessible points, Obstacles in chain survey.

2. Compass surveying
Measurement of bearings, Preparation of map, Distance between two inaccessible points by chain and compass, Chain and compass traverse.

3. Plane Table survey
Temporary adjustments of plane table and Radiation method, Intersection, Traversing and Resection methods of plane tabling, Three-point problem.

4. Levelling
Temporary adjustment of Dumpy level, Differential levelling, Profile levelling and plotting the profile, Longitudinal and cross sectioning, Gradient of line and setting out grades, Sensitiveness of Bubble tube.

5. Contouring
Direct contouring, Indirect contouring – Block levelling, Indirect contouring – Radial contouring, Demonstration of minor instruments.

Text Book:
1. Surveying Volume I And Volume II By B.C.PunmiaBy Laxmi Publications Limited.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR
Lab Manual

PRACTICAL NO: 1

Objective: To measure bearings of a closed traverse by prismatic compass and to adjust the traverse by graphical method.

Apparatus: Prismatic compass, pegs, ranging rods etc

Theory: Definition: Surveying which involves series of connected lines is known as traversing The sides of traverse are known is known as Traverse legs.

In traversing with a compass free or loose needle method is employed to determine direction of survey line. The compass is setup at each of the successive station and fore & back bearing of each line is determined All the readings are noted in field book. Each of the line is observed independently & errors are calculated, compensated. The field work consists of primary survey, marking of stations, running of traverse lines.

Traverse stations should be selected that

I) They are visible from each other

II) They are as long as possible.

III) The line joining them are as near the boundaries & objects To be located as possible

Procedure:

Let us say we have to run a closed compass traverse ABCDEA. Set the prismatic compass at point A. center it and level it.

1. Take bearings of traverse lines AB and AE.

2. Shift the compass to point B center it and level it. Take the bearings BC and BA.

3. Link-wise complete the traverse as shown in fig (a).

4. Measure the length of traverse line AB, BC, CD, DE, and EA.

5. Record the observation in tabular columns.

6. Care must be taken to see that the stations are not affected by local attractions. If they are affected corrections to local attractions should be applied first and Then the traverse should be plotted with corrected bearings.

7. Simplest method of plotting is angle and distance method with a protractor. If Last point is falling short by some distance in meeting the first point then it means that there is a closing error.

So, traverse should be adjusted by “Bow ditch’s graphical method”

PRACTICAL NO: 2

OBJECTIVE: To find out reduced levels of given points using dumpy/Auto level.

Apparatus:- Dumpy level, tripod stand, leveling staff etc.

Theory:-The dumpy level is a simple and compact instrument; the telescope is rigidly fixed to its supports it can be rotated about its longitudinal axis. Level tube is permanently placed so that axis lies in same vertical plane. A focusing screw near the eye piece provided to get clear image of the object and to Bisect cross hair.

Reduction of levels

H.I method:-

The reduced level of the line of collimation is said to be the height of the instrument. In this system height of the line of collimation is found out by adding back side reading to the R.L of bench mark on which BS is taken. Then RL of intermediate points and the change point are obtained by subtracting the respective staff reading from the height of instrument (HI). To find new HI of change point BS is taken on last point.

Procedure:

Let A and B be the two given points whose difference is elevation is to be found.

Set the level at convenient point carryout temporary adjustments and take B.S on A

1. Take FS on the Point C

1. Shift the instrument to point O2 and perform temporary adjustments.

2. Take B.S on C.

3. Take F.S. on D.

4. Shift the instrument to point O3 and perform temporary adjustments.

5. Take B.S on D

6. Take F.S on B.

7. Find the difference in elevation between A and B by both the methods.

PRACTICAL NO: 3

Objective: To perform fly leveling with an Auto /tilting level.

Apparatus: Dumpy level, Telescope Staff, Tripod.

Theory: Fly leveling is done to calculate RL of a particular point from the known bench mark e.g. in fig showing R.L of particular point is A is calculated taking back sight on BM & F.s on A.

PROCEDURE:

1. Set up the level on the tripod at a convenient height and bring the foot screws approximately to the middle of its rim.

2 .By temporary adjustments bring the bubble at centre open out typical leveling field book columns.

3. Sight the given points and take the staff reading and note down the readings at the appropriate columns.

4. If there are any points for away and is not clearly visible take. A change point and the leveling is continued.

5. After finishing the leveling, calculate the elevations by the rise and fall method and apply necessary checks.

PRACTICAL NO: 4

Objective: To measure horizontal angle by method of reiteration

Apparatus: Theodolite, ranging rods and arrows.

Theory: Reiteration is a method of measuring horizontal angles with high precision. It is less

tedious and is generally preferred when there are several angles to be measured at a station. Several angles are measured successively and finally the horizon is closed. Closing the horizon is the process of measuring the angles around a point to obtain a check on their sum which should be equal to 3600.

Procedure:

1. Select a station point O.

2. Set the theodolite at O and do the temporary adjustments. The telescope is adjusted for right face right swing.

3. Set the vernier A to zero using upper clamp. Loosen the lower clamp, direct the telescope to the station point A and bisect A exactly by using the lower clamp and lower tangent screw.

4. Note the vernier readings (A and B).

5. Loosen the upper clamp and turn the telescope clockwise until the point B is exactly bisected.

6. Note the vernier readings (A and B).

7. The mean of the two vernier readings gives the value of <AOB.

8. Bisect all the points successively and note the readings of both verniers at each bisection.

9. Finally close the horizon by sighting the station point A. The A vernier The A vernier should be 3600. If not, note the closing error.

10. Adjust the telescope for left face left swing.

11. Repeat the whole process by turning the telescope in anticlockwise direction.

12. Distribute the closing error proportionately the several observed angles.

13. Take the average of face left and face right observations to give the corresponding horizontal angles.

PRACTICAL NO 5:

Objective: To measure the horizontal angle AOB by repetition method.

Apparatus: Theodolite, ranging rods and arrows.

Theory: The method of repetition is used to measure a horizontal angle to a finer degree of accuracy. By this method, an angle is measured two or more times by allowing the vernier to remain clamped each time at the end of each measurement instead of setting it back at zero when sighting at the previous station. Thus an angle reading is mechanically added several times depending upon the number of repetitions. The average horizontal angle is then obtained by dividing the final reading by the number of repetitions. For very accurate work the method of repetition is used.

Procedure :

1. Select a station point O.

2. Set the theodolite at O and do the temporary adjustments. The telescope is adjusted for right face right swing.

3. Set the vernier A to zero using upper clamp. Loosen the lower clamp, direct the telescope to the station point A and bisect A exactly by using the lower clamp and lower tangent screw.

4. Note the vernier readings (A and B).

5. Loosen the upper clamp and turn the telescope clockwise until the point B is exactly bisected.

6. Note the vernier readings (A and B).

7. The mean of the two vernier readings gives gives the value of <AOB.

8. Loosen the lower clamp and turn the telescope to station point A and bisected A by using the lower clamp and lower tangent screw.

9. Loosen the upper clamp and turn the telescope clockwise until the point B is exactly bisected. Now the vernier reading is twice the value of the angle.

10. Repeat the process for the required number of times (usually 3).

11. The correct value of the angle AOB is obtained by dividing the final reading by the number of repetition.

12. Adjust the telescope for left face left swing.

PRACTICAL NO 6:

Objective - To measure direct angle, deflection angle and magnetic bearing of line by using theodolite

Apparatus: Transit theodolite, ranging rod, peg etc

Procedure:-Set up the theodolite at O and level it accurately set vernier A to 0°0′0″.Loose the lower plate and take back sight on A.

1. Loose upper plate rotate telescope clockwise and bisect B exactly read both vernier.

2. Plunge the telescope turns the instrument about its outer axis and take back sight on A the reading on vernier A will be same as in Step 1.

3. Loose the upper plate, turn the telescope clockwise and again bisect B exactly.

4. Read both vernier. The reading will be twice the previous, <AOB will be obtain by dividing the final reading by 2.

PRACTICAL NO: 7

Objective: - To measure vertical angle between two points using theodolite.

Apparatus: Transit theodolite Tripod, ranging rod, pegs etc.

Procedure: To measure the vertical angle of an object P

1. Set up the instrument over station O and level it carefully with respect to altitude bubble.

2. By means of vertical circle clamp and tangent screw, set 0 of the vertical circle exactly to 0 of the circle.

3. Bring the bubble of the altitude level to the centre of its run by means of foot &clip screw.

4. The line of sight is thus made horizontal.

5. Loose the vertical circle clamp and direct the telescope in vertical plane towards the object P, and bisect exactly using vertical tangent screw.

6. Read both the vernier C and D, the mean of two readings gives angle for that face.

7. Change the face and repeat the above process, and get the face reading.

8. The average of two face values gives exact value of required vertical angle.

PRACTICAL NO 8:

Objective: Setting out of simple circular curve by Rankine method of tangential angle.

Apparatus: Theodolite, ranging rods, pegs, arrows etc.

Theory: A deflection angle to any point on the curve is the angle at P.C between the back tangent and the chord from the P C to that point.

Theory:

T1V= rear tangent

T1 = Point to curve

= the tangential angles or the angles with each of the successive chords

T1A, AB, BC etc. Makes with the respective tangents to the curve at T1, A, B etc

= Total tangential angles of the deflection angles to the points A , B, C etc.

C1, C2, C3 = lengths f the chords T1A, AB, BC etc...

A1A = tangent to the curve at A

= 1719 C /R minutes

For the first chord

= tangential angle for the chord AB

Hence, the deflection angle for any chord is equal to the deflection angle for the previous chord plus the tangential angle for that chord.

Procedure:

1. Set the theodolite at the point of curve T1.

2. With both the plates clamped to zero, direct the theodolite to bisect the point of intersection V. The line of sight is thus in the direction of the rear tangent.

3. Release the vernier plate and set angle 1 on the vernier .The line of sight is thus directed along chord T1A.

4. With zero end of tape pointed at T1 and arrow held at a distance T1A = c along it, swing the tape around T1 till the arrow is bisected by the cross hairs.

5. Thus the first point A is fixed.

6. Set the second deflection angle 2 on the vernier so that the line of sight is directed along T1B.

7. with the zero end of the tape pinned at A, and an arrow held at distance AB = C along it, swing the tape around A till the arrow is bisected by the cross hairs, thus fixing the point B.

8. Repeat steps 4 and 5 till last point is reached.

Result: The simple curve was set by Rankine's method of tangential angle.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Department Of Civil Engineering

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

Title of Course: Building Design & Drawing Lab
Course Code: CE393
L-T-P scheme: 0-0-3 **Course Credit: 2**

Objectives:
1. The students will be able to draw efficiently the line diagram, plan, elevation and sectional drawings for buildings.
2. The students will have a clear knowledge of the details of reinforcements for a RCC staircase.
3. The students will develop a fundamental concept of various types of foundation and basic concepts of pile foundation.

Learning Outcomes: The students will have a clear understanding of the various structural components of the building and the concept of scale factor to the actual ratio of drawing proportionately various components. The students will have a clear idea of the types of foundation and the types of footing for a RCC column and develop a understanding of the pile foundation. The students will be able to draw the line diagram, plan, elevation and sectional drawings of the following: Residential Buildings, Office Buildings and Schools. The students will also develop a understanding of the types of roof trusses, RCC roof with details of reinforcements and King Post and Queen Post trusses. They will have a clear knowledge of the proportioning and design of stair cases and details of reinforcements for RCC staircase.

Course Contents:
Exercises that must be done in this course are listed below:
1. Foundations
Spread foundation for walls and columns; Footing for a RCC column, raft and pile foundations.

2. Doors and Windows
Glazed and paneled doors of standard sizes; Glazed and paneled windows of standard sizes; special windows and ventilators.

3. Stairs
Proportioning and design of a dog-legged, open well RCC stair case for an office / Residential building; Details of reinforcements for RCC stair cases; Plan and elevation of straight run, quarter turn, dog-legged and open well stair cases.

4. Roofs and Trusses
Types of sloping roof, lean-to roofs, RCC roof with details of reinforcements, King post and Queen post trusses.

5. Functional Design of Buildings
To draw the line diagram, plan, elevation and section of the following: Residential Buildings (flat, pitched and combined roofs), Office Buildings (flat roof), School .The designs must show positions of various components including lift well and their sizes. Introduction to drawing by using software package.

Text Book:
1. Principles of Building Drawing Shah & Kale.
2. Text Book of Building Construction Sharma &Kaul.
3. Building Construction B C Punmia.