










































CSL 201	DATA STRUCTURES LAB	CATEGORY	L	T	P	CREDIT	YEAR OF INTRODUCTION
		PCC	0	0	3	2	2019

**Preamble:** The aim of the Course is to give hands-on experience for Learners on creating and using different Data Structures. Data Structures are used to process data and arrange data in different formats for many applications. The most commonly performed operations on data structures are traversing, searching, inserting, deleting and few special operations like merging and sorting.

**Prerequisite:** Topics covered under the course Programming in C (EST 102)

CO1	Write a time/space efficient program using arrays/linked lists/trees/graphs to provide necessary functionalities meeting a given set of user requirements ( <b>Cognitive Knowledge Level: Analyse</b> )
CO2	Write a time/space efficient program to sort a list of records based on a given key in the record ( <b>Cognitive Knowledge Level: Apply</b> )
CO3	Examine a given Data Structure to determine its space complexity and time complexities of operations on it ( <b>Cognitive Knowledge Level: Apply</b> )
CO4	Design and implement an efficient data structure to represent given data ( <b>Cognitive Knowledge Level: Apply</b> )
CO5	Write a time/space efficient program to convert an arithmetic expression from one notation to another ( <b>Cognitive Knowledge Level: Apply</b> )
CO6	Write a program using linked lists to simulate Memory Allocation and Garbage Collection ( <b>Cognitive Knowledge Level: Apply</b> )

### Mapping of course outcomes with program outcomes

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1												
CO2												
CO3												
CO4												
CO5												
CO6												

Abstract POs defined by National Board of Accreditation			
PO#	Broad PO	PO#	Broad PO
PO1	Engineering Knowledge	PO7	Environment and Sustainability
PO2	Problem Analysis	PO8	Ethics
PO3	Design/Development of solutions	PO9	Individual and team work
PO4	Conduct investigations of complex problems	PO10	Communication
PO5	Modern tool usage	PO11	Project Management and Finance
PO6	The Engineer and Society	PO12	Life long learning

### Assessment Pattern

Bloom's Category	Continuous Assessment Test (Internal Exam) <i>Percentage</i>	End Semester Examination <i>Percentage</i>
Remember	20	20
Understand	20	20
Apply	60	60
Analyse		
Evaluate		
Create		

### Mark Distribution

Total Marks	CIE Marks	ESE Marks	ESE Duration
150	75	75	3 hours

### Continuous Internal Evaluation Pattern:

Attendance : 15 marks

Continuous Evaluation in Lab : 30 marks

Continuous Assessment Test : 15 marks

Viva-voce : 15 marks

**Internal Examination Pattern:** The marks will be distributed as Algorithm 30 marks, Program 20 marks, Output 20 marks and Viva 30 marks. Total 100 marks which will be converted out of 15 while calculating Internal Evaluation marks.

**End Semester Examination Pattern:** The marks will be distributed as Algorithm 30 marks, Program 20 marks, Output 20 marks and Viva 30 marks. Total 100 marks will be converted out of 75 for End Semester Examination.

**Operating System to Use in Lab** : Linux

**Compiler/Software to Use in Lab** : gcc

**Programming Language to Use in Lab** : Ansi C

**Fair Lab Record:**

All Students attending the Data Structures Lab should have a Fair Record. The fair record should be produced in the University Lab Examination. Every experiment conducted in the lab should be noted in the fair record. For every experiment in the fair record the right hand page should contain Experiment Heading, Experiment Number, Date of Experiment, Aim of Experiment, Data Structure used and the operations performed on them, Details of Experiment including algorithm and Result of Experiment. The left hand page should contain a print out of the code used for the experiment and sample output obtained for a set of input.

**SYLLABUS**

1. Implementation of Polynomials and Sparse matrices using arrays\*\*
2. Implementation of Stack , Queues, Priority Queues, DEQUEUE and Circular Queues using arrays\*\*
3. Application problems using stacks: Conversion of expression from one notation to another notation . \*\*
4. Implementation of various linked list operations. \*\*
5. Implementation of stack, queue and their applications using linked list.pression
6. Implementation of trees using linked list
7. Representation of polynomials using linked list, addition and multiplication of polynomials. \*\*
8. Implementation of binary trees using linked lists and arrays- creations, insertion, deletion and traversal. \*\*
9. Implementation of binary search trees – creation, insertion, deletion, search
10. Any application programs using trees
11. Implementation of sorting algorithms – bubble, insertion, selection, quick, merge sort

and heap sort.\*\*

12. Implementation of searching algorithms – linear search, binary search.\*\*

13. Representation of graphs and computing various parameters (in degree, out degree etc.) - adjacency list, adjacency matrix.

14. Implementation of BFS and DFS for each graph representations.\*\*

15. Implementation of hash table using your own mapping functions and observe collisions and overflow resolving schemes.\*\*

16. Simulation of first-fit, best-fit and worst-fit allocations.

17. Simulation of a basic memory allocator and garbage collector using doubly linked list.  
\*\* mandatory.

### **DATA STRUCTURES LAB - PRACTICE QUESTIONS**

1. Write a program to read two polynomials and store them in an array. Calculate the sum of the two polynomials and display the first polynomial, second polynomial and the resultant polynomial.
2. C Write a program to enter two matrices in normal form . Write a function to convert two matrices to tuple form and display it. Also find the transpose of the two matrices represented in tuple form and display it. Find the sum of the two matrices in tuple form and display the sum in tuple form.
3. Write a program to enter two matrices in normal form . Write a function to convert two matrices to tuple form and display it. Also find the transpose of the two matrices represented in tuple form and display it. Find the sum of the two matrices in tuple form and display the sum in tuple form.
4. Implement a circular queue using arrays with the operations:
  - 4.1. Insert an element to the queue.
  - 4.2. Delete an elements from the queue.
  - 4.3. Display the contents of the queue after each operation.
5. Implement a Queue using arrays with the operations:

- 5.1. Insert elements to the Queue.
- 5.2. Delete elements from the Queue.
- 5.3. Display the contents of the Queue after each operation.
6. Implement a Stack using arrays with the operations:
  - 6.1. Pushing elements to the Stack.
  - 6.2. Popping elements from the Stack
  - 6.3. Display the contents of the Stack after each operation.
7. Implement a Priority Queue using arrays with the operations:
  - 7.1. Insert elements to the Priority Queue.
  - 7.2. Delete elements from the Priority Queue.
  - 7.3. Display the contents of the Priority Queue after each operation.
8. Implement a Double-Ended Queue (DEQUEUE) with the operations:
  - 8.1. Insert elements to the Front of the queue.
  - 8.2. Insert elements to the Rear of the queue
  - 8.3. Delete elements from the Front of the queue.
  - 8.4. Delete elements from the Rear of the queue.
  - 8.5. Display the queue after each operation.
9. Using stack convert an infix expression to a postfix expression and evaluate the postfix expression.
10. Write a program to convert an infix expression to a prefix expression using stacks.
11. Convert an infix expression to a postfix expression without using a stack
12. Write a menu driven program for performing the following operations on a Linked List:
  - 12.1. Display
  - 12.2. Insert at Beginning
  - 12.3. Insert at End
  - 12.4. Insert at a specified Position
  - 12.5. Delete from Beginning
  - 12.6. Delete from End
  - 12.7. Delete from a specified Position
13. Implement a stack using linked list with the operations:
  - 13.1. Push elements to the queue.
  - 13.2. Pop elements from the queue.
  - 13.3. Display the queue after each operation.
14. Implement a Queue using linked list with the operations:

- 14.1. Insert an element to the queue.
  - 14.2. Delete an element from the queue.
  - 14.3. Display the queue after each operation.
15. Write a program to reverse the content of queue using stack
  16. Write a program to read two polynomials and store them using linked list. Calculate the sum of the two polynomials and display the first polynomial, second polynomial and the resultant polynomial.
  17. Write a program to read two polynomials and store them using linked list. Find the product of two polynomials and store the result using linked list. Display the resultant polynomial.
  18. Write a program for addition of polynomials containing two variables using linked list.
  19. The details of students (number, name, total-mark) are to be stored in a linked list. Write functions for the following operations:
    - 19.1. Insert
    - 19.2. Delete
    - 19.3. Search
    - 19.4. Sort on the basis of number
    - 19.5. Display the resultant list after every operation
  20. Create a Doubly Linked List from a string taking each character from the string. Check if the given string is palindrome in an efficient method.
  21. Create a binary tree with the following operations
    - 21.1. Insert a new node
    - 21.2. Inorder traversal.
    - 21.3. Preorder traversal.
    - 21.4. Postorder traversal.
    - 21.5. Delete a node.
  22. Write a program to create a binary search tree and find the number of leaf nodes
  23. Create a binary search tree with the following operations:
    - 23.1. Insert a new node .
    - 23.2. Inorder traversal.
    - 23.3. Preorder traversal.
    - 23.4. Postorder traversal.
    - 23.5. Delete a node.

24. Write a program to sort a set of numbers using a binary tree.
25. Represent any given graph and
- 25.1. Perform a depth first search .
  - 25.2. Perform a breadth first search
26. Create a text file containing the name, height, weight of the students in a class. Perform Quick sort and Merge sort on this data and store the resultant data in two separate files. Also write the time taken by the two sorting methods into the respective files.

Eg.            Sony Mathew            5.5      60  
                 Arun Sajeev            5.7      58  
                 Rajesh Kumar           6.1      70

27. Write a program to sort a set of numbers using Heap sort and find a particular number from the sorted set using Binary Search.
28. Implement a Hash table using Chaining method. Let the size of hash table be 10 so that the index varies from 0 to 9.
29. Implement a Hash table that uses Linear Probing for collision resolution