14/C

Course Title: Data Structures and Algorithms Lab			
Type of Course: DCC	**Level of Course:** 4.5	**Delivery Sub Type of the course:** Practical	
Course code: CSE-3-202-P	**No. of credits:** 2	**T-P-S:** 0-4-0	**Learning hours:** 60
Pre-requisite and Co-requisite of Course: Computer Programming Knowledge (C/C++)			
Department: Computer Science Engineering			

Syllabus:

Module 1: Introduction to Data Types and Arrays
Module 2: Efficiency of Algorithms - Time and Space Complexity
Module 3: Sorting Algorithms
Module 4: Linked Lists
Module 5: Stacks
Module 6: Queues
Module 7: Hash Tables
Module 8: Trees
Module 9: Graphs

Course objectives

- Understand and apply fundamental data structures and algorithms.
- Implement static and dynamic representations of data structures.
- Analyze the time and space complexity of algorithms.
- Develop proficiency in tree and graph algorithms for problem-solving.
- Use advanced data structures for specialized applications.
- Enhance problem-solving skills through efficient algorithm design.

Course content

Module / Unit	Topic	T	P	S
1	• Write a program for basic array operations like insertion, deletion, updating, traversal on Array. • Write a program to create a 2D array using pointers and perform matrix operations (addition, transpose).		2	
2	• Implement recursive vs. iterative solutions for problems like Fibonacci series, factorial calculation, and analyze their time/space complexity. • Write programs to compare linear vs. binary search and measure their time complexity in terms of input size. • For linear search, binary search, and bubble sort, write test cases that illustrate best, worst, and average cases, and compare the results.		4	
3	• Write a program to implement insertion sort. • Write a program to implement selection sort. • Write a program to implement merge sort. • Write a program to implement quick sort with randomized pivot selection. • Write a program to implement heap sort. • Write a program to implement radix and/or shell sort.		6	
4	• Write programs to create a singly linked list with nodes containing integer data, with functions for insertions, deletions, updating, traversal. • Write programs to create a doubly linked list with nodes containing integer data, with functions for insertions, deletions, updating, traversal. • Write a program to reverse a singly linked list using pointers. • Write a program to merge two sorted linked lists into a single sorted linked list.		8	

14/11/25

5	• Write a program to implement a stack using a static array and perform basic stack operations like push, pop, etc. • Write a program to implement a stack using a dynamic linked list and perform basic stack operations like push, pop, etc. • Write a program to evaluate an expression written in postfix notation (Reverse Polish Notation, RPN) using a stack. • Write a program to evaluate an expression written in prefix notation (Polish Notation) using a stack. • Write a program to convert an infix expression to postfix notation (e.g., A + B * C to A B C * +). • Write a program to convert an infix expression to prefix notation.		8	
6	• Write a program to implement a basic queue using an array, and perform basic operations like enqueue, dequeue, etc. • Write a program to implement a basic queue using a linked list, and perform basic operations like enqueue, dequeue, etc. • Write a program to implement a circular queue and perform basic operations like enqueue, dequeue, etc. • Write a program to simulate a simple task scheduler using a priority queue, where each task has an ID and duration.		8	
7	• Write a program to implement a simple hash table using an array and hash function to map keys to indices. Write functions to insert and search for elements in the hash table using modulo-based hashing. • Write a program to handle collisions in a hash table using linear probing. • Write a program to use a hash table to count the frequency of each element in an array of integers.		6	
8	• Write a program to create a binary tree and implement basic operations like, insert, delete, search. • Write a program to perform pre-order, in-order, and post-order traversal on a binary tree and print the elements in each traversal order. • Write a program to implement a binary search tree (BST) with functions for insertion, deletion, and searching. • Write a program to implement an AVL tree with functions to perform insertion, deletion, and rotations (single and double rotations).		10	
9	• Write a program to represent a graph using an adjacency matrix or adjacency list. • Write a program to implement breadth-first search (BFS) on a graph and print the order of visited nodes. • Write a program to implement depth-first search (DFS) on a graph and print the order of visited nodes.		8	

Scheme of End Semester Examination **Total: 100 marks**
As per Regulation 2A (30% CA + 70% EoSE)

Recommended Books and References:

- **E. Horowitz & Sahni, Fundamental Data Structure, Galgotia Book Source, 1983.**
- **Tannenbaum, Data Structure Using C, Pearson Education, 2003.**
- **Kruz, Data Structure and Programming Design, 1987.**
- **N. Wirth, Algorithms +Data Structure = Program, Prentice Hall of India, 1979.**
- **Data Structures through C, Yashawant Kanetkar, BPB Publications, 4th Ed., 2022.**

Learning outcomes: Upon Completing the Course, Students will able to:

- **Demonstrate proficiency in implementing core data structures (e.g., linked lists, stacks, queues, hash tables, trees, graphs) in C++/C.**
- **Apply appropriate algorithms for sorting, searching, and traversing data structures.**
- **Analyze and compare algorithm efficiency using time and space complexity.**
- **Utilize advanced data structures like heaps, AVL trees, and B+ trees in problem-solving.**
- **Implement graph algorithms (BFS, DFS, shortest path) for practical applications.**
- **Develop optimized solutions using best-case, worst-case, and average-case complexity analysis.**
- **Use the appropriate data structures in context of a solution to a given problem.**

14/1/25