CS 202: DATA STRUCTURES (3-1-2: 5)

Introduction & Overview: Concept of data type, definition and brief description of various data structures, operations on data structures, algorithm complexity, Big Oh notation, recursion, some illustrative examples of recursive functions.

Review of Pointers and Dynamic Memory Management: Understanding pointers, usage of pointers, memory management functions, debugging pointers.

Arrays: Linear and multi-dimensional arrays and their representation, operations on arrays, sparse matrices and their storage.

Linked Lists: Linear linked list, operations on linear linked list, doubly linked list, operations on doubly linked list, application of linked lists.

Stacks: Sequential and linked representations, operations on stacks, multi stacks, application of stacks such as parenthesis checker, evaluation of postfix expressions, conversion from infix to postfix representation, implementing recursive functions.

Queues: Sequential representation of queue, linear queue, circular queue, operations on linear and circular queue, linked representation of a queue and operations on it, priority queues, applications of queues.

Trees: Basic terminology, array and linked representations of trees, traversing a binary tree using recursive and non-recursive procedures, inserting a new node, deleting a node, counting nodes, finding height, finding a mirror image of a binary tree, threaded binary trees, AVL trees and B-trees.

Graphs: Basic terminology, representation of graphs (adjacency matrix, adjacency list), traversal of a graph (breadth first search and depth-first search), adding nodes, deleting nodes, applications of graphs in problems such as finding shortest paths, obtaining minimum cost spanning tree, etc.

Hashing: Comparing direct address tables with hash tables, hash functions, concept of collision and its resolution using open addressing and separate chaining, double hashing, rehashing.

Sorting & Searching: Sorting arrays using bubble sort, selection sort, insertion sort, quick sort, merge sort, heap sort, shell sort, tree sort, radix sort, etc., searching an element using linear search and binary search techniques, concatenation of arrays and merging sorted arrays.

Heaps: Representing a heap in memory, operations on heaps, application of heap in implementing priority queue and heapsort algorithm.

Suggested List of Laboratory Experiments

1. Implement an algorithm to insert an element at any arbitrary position in an array of integer numbers and also implement an algorithm to display the condition of the array before and after insertion.
2. Implement an algorithm to delete an element in an array of integer numbers and also implement an algorithm to display the condition of the array before and after deletion.
3. Implement an algorithm to reverse the elements of an array of integer numbers and also implement an algorithm to display the condition of the array before and after reversal.
4. Implement an algorithm to create a linked list, containing integer data and also implement an algorithm to display the nodes of the linked list. Show the output w.r.t an input of a set of 6.
5. Write a program for addition of two polynomial using linked list.
6. Write a program for multiplication of two polynomial using linked list.
7. Write a C program to implement sorting of **n** numbers using Bubble sort.
8. Write a C program to implement sorting of **n** numbers using Selection sort
9. Write a C program to implement sorting of **n** numbers using Insertion sort.
10. Write a C program to implement sorting of **n** numbers using Quick sort.
11. Write a C program to implement sorting of **n** numbers using Merge sort.
12. Write a C program to implement searching of a **key** from **n** numbers (given in Descending order) using Binary search.
13. Write a C program to find a **key** from **n** numbers using sequential search (Linear search)
    & if found, show the position.

14. Implement algorithms to insert an element in a stack(push), to delete an element from a stack(pop) and to display the elements of the stack.[Assume: initially, top= -1]
15. Implement algorithms to insert an element in a queue, to delete an element from a queue and to display the elements of the queue.[Assume: initially, front= -1, rear= -1]
16. Implement algorithms to insert an element in a circular queue, to delete an element from a circular queue and to display the elements of the circular queue.[Assume: initially, front= 0, rear= -1]
17. Write a C program to solve Tower of Hanoi problem for n disks.
18. Write a C program to generate **n** Fibonacci numbers using both recursive and non-recursive methods.
19. Implement a binary tree using array.
20. Implement a binary search tree using linked list and traverse in pre- order, in-order and post-order.
21. Create a binary search tree of **N** nodes with given **N** elements and search a given key element.
22. Write a C program to implement sorting of **n** numbers using binary search tree.
23. Implement an AVL tree.
24. Create a Hash table to store the account number and balance of the customers. Provide proper option to create, search and delete customer details.
25. Write a c program to create a file, named “StudentDatabase” . Store the the name, roll number, phone number and average marks of **N** students, where **N** is a natural number between 2 to 10.

    Ex:

Sl.No.	**Name**	**roll number**	**phone number**	**average marks**
1.	xyz	1234567	9900221188	8.2

    After creating database, modify the phone no. and marks of ith student, 1< i < =**N**

Text Books:

1. Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman, Data Structures & Algorithms, Pearson.
2. Yedidyah Langsam, Aaron M. Tenenbaum, Moshe J. Augenstein, Data Structures using C and C++, PHI Learning.