Data Structure and Algorithms Laboratory Exercise

20 %

LAB Objectives: The objective of this lab is to teach students various data structures and to explain them algorithms for performing various operations on these data structures. This lab complements the data structures course. Students will gain practical knowledge by writing and executing programs in various data structures such as arrays, linked lists, stacks, queues, trees, graphs, hash tables and search trees.

This lab will enable the students to:

- Understand various data representation techniques in the real world.
- Implement linear and non-linear data structures.
- Analyze various algorithms based on their time and space complexity.
- Develop real-time applications using suitable data structure.
- Identify suitable data structure to solve various computing problems.

Programming language: Python

The followings are needed to write in the Lab notebook for submission.

1) Aim
2) Algorithm
3) Program
4) Input
5) Output step by step (i.e., each pass)
6) Explanation of variable used.
7) Learning outcome

ADMINISTRATIVE POLICY OF THE LABORATORY

- Students must perform class assessment tasks individually without help of others.
- Viva/writing examination for programs will be taken and considered as a performance.
- Plagiarism is strictly forbidden and will be dealt with punishment.

MID TERM EXAMINATION

There will be a 40-minutes written mid-term examination. Different types of questions will be included such as MCQ, writing code fragments etc.

FINAL TERM EXAMINATION

There will be a one-hour written examination. Different types of questions will be included such as MCQ, writing programs etc.

Lab sessions

Week 1:

No. 01 - Practice Problems: Hands on Array

- **A.** Implement 2D Array, matrix, matrix multiplication.
- **B.** Implement effective way of sparse matrix representation and find transpose of a Sparse Matrix representation.
- **C.** Implement To store a polynomial expression using array.
- **D.** Write a program for implementing the following techniques using array.
  - i. Maximum number in an array,
  - ii. Linear search, and
  - iii. Fibonacci series

Week 2

No. 02 - Practice Problems: Searching and Sorting

- **A.** Take a randomly generated array and sort it is using bubble sort. Then find an element using binary search.
- **B.** Implementation of the “Insertion, Selection Sort & Binary Search” Algorithm to create a program in C/python.

Optional/ Take Home:

- **C.** Take a randomly generated array and sort it using insertion sort. Then find an element using binary search.
- **D.** Take a randomly generated array and sort it using selection sort. Then find an element using binary search.
- **E.** Take a randomly generated array and reverse-sort it using insertion sort. Then find an element using binary search.
- **F.** Take a randomly generated array and reverse-sort it is using bubble sort. Then find an element using linear search.
- **G.** Take a randomly generated array and sort the 2nd half using bubble sort. Then find the median using linear search.

No. 03 - Practice Problems: Merge Sort and Quick Sort

A. Write a program for implementing the following sorting techniques to arrange a list of integers in ascending order. Also demonstrate a comparison table (Data movement and Data Comparison) of different Sorting Algorithms.

   a. Quick sort
   b. Merge sort

Optional/ Take Home:

B. Find median of an array using quicksort. You cannot sort the entire array. Additionally, if it is evident that median is in a particular half of the array, you must stop sorting the other half of the array.

Week 3:

No. 04 - Practice Problems: Implementation of Stack and Queue

A. Implementation of the "Stack using an array". Use these concept: Push(value), Pop(), FULL, EMPTY, and FindTOP position.

B. Implementation of the "Queue using an array". Use these concept: Enqueue, Dequeue, Front, Rear, isfull() and isempty()

C. Implementation of the "Circular Queue using an array". Use these concept: Enqueue, Dequeue, Front, Rear, isfull() and isempty()

Optional/ Take Home:

D. Implement a function popmin() in a stack that pops the minimum value in a stack.

E. Implement a function Dequeuemin() in a queue that pops the minimum value in a queue.

F. Implement stack using queue and queue using stack.

No. 05 - Practice Problems: Application of Stack

A. Implementation of the "Tower of Hanoi (Recursive and non-Recursive)" Algorithms to create a program in C/Python. And Print tower of Hanoi steps.

Optional/ Take Home:

B. Implementation of the "Infix to Postfix format" Algorithm to create a program in C/Python.

C. Implementation of the "Postfix Evaluation" Algorithm to create a program in C/Python.

No. 06 - Practice Problems: Linked Lists

A. Implementation of the "Single Linked List" Algorithm to create a program in C/Python. Perform the following operations:

i. Insert at beginning
ii. Insert at end
iii. Insert at any position
iv. Delete at beginning
v. Delete at end
vi. Delete at any position

Optional/ Take Home:

B. Print the contents of a simple linked list in reverse order using recursion.
C. Merge two sorted Linked List.

Week 4:

No. 07 - Practice Problems: Doubly and Circular Linked Lists

A. Insert into (at beginning, at any position and at end) and delete (beginning, at any position and at end) from a Doubly Linked List

Optional/ Take Home:

B. Insert into (beginning, at any position and at end), and delete (at beginning, at any position and at end) from a Circular Linked List
C. Convert Singly Linked List into Doubly Linked List
D. Reverse a Doubly Linked List

No. 08 - Practice Problems: Stack and Queue using Linked Lists

Optional/ Take Home:

A. Implementation of the stack and queue using Linked List
B. Implement a function popmin()/dequemin() in a stack/queue that pops the minimum value in a stack/queque.

No. 09 - Practice Problems: Tree and its Traversal

Implementation of the "Binary Tree" Algorithms to create a program in C/Python. Also, a program to implement "Pre Order, In Order, Post Order Traversal of Binary Tree (Recursive & Non-Recursive- Not mandatory)" Algorithms.

Week 5 :

No. 10 - Practice Problems: Binary Search Tree

A. Implementation of the “Binary Search Tree & insert, search, finding successor of BST” Algorithms to create a program in C/Python.

Optional/ Take Home:

B. Finding number of duplicate entries in a BST and use tree traversal.

No. 11 - Practice Problems: Hashing

A. Implementation of the “Hash Function” (using array to insert and search). Also one (any) “Collision resolution Method” Algorithms to create a program in C/Python.

Optional/ Take Home:

B. Implement a hashing module in an array. Use modulo value as hash function.

No. 12 - Practice Problems: Breadth-First Search and Depth-First Search

Optional/ Take Home:

A. Implementation of the “Breadth First Search & Depth First Search” Algorithms to create a program in C/Python.
B. Finding height of a tree using DFS.
C. Finding number of nodes in each level of a tree using BFS.
D. Finding total number of nodes in a tree using BFS.