Pandit Deendayal Energy University

School of Technology

Department of Computer Science & Engineering

Odd Semester 2021-2022

LAB MANUAL FOR DATA STRUCTURES

BY

DR. ADITYA SHASTRI

Practical No. 1 [Revision of Arrays]

Programs covering basics of Arrays: searching, sorting, minimum and maximum elements etc.

1. Write a program in C to perform linear and binary search.

2. Write a program in C to perform bubble sort, insertion sort and selection sort. Take the array size and array elements from user.

3. Write a program in C that obtains the minimum and maximum element from the array. Modify this program to give the second largest and second smallest element of the array.

Practical No. 2 [Revision of Structures]

Programs covering structure definition, array of structure, nested structures etc.

1. Create a structure Student in C with student name, student roll number and student address as its data members. Create the variable of type student and print the values.

2. Modify the above program to implement arrays of structure. Create an array of 5 students and print their values.

3. Create a structure Organization with organization name and organization ID as its data members. Next, create another structure Employee that is nested in structure Organization with employee ID, employee salary and employee name as its data members. Write a program in such a way that there are two organizations and each of these contains two employees.

Practical No. 3 [Revision of Pointers]

Programs covering use of pointers with arrays, structures, functions etc.

1. Write a program in C to implement arrays of pointers and pointers to arrays.

2. Write a program in C to implement pointers to structures.

3. Write a program in C to perform swapping of two numbers by passing addresses of the variables to the functions.

Practical No. 4 [Linked List]

1. Write a program that takes ***two sorted lists*** as inputs and merge them into one sorted list.
   For example, if the first linked list ***A*** is 5 =>10 =>15, and the other linked list ***B*** is 2 => 3 => 20, then output should be 2 => 3 => 5 => 10 => 15 => 20.

2. Write a program to ***insert a new node*** into the linked list. A node can be added into the linked list using three ways: [Write code for all the three ways.]
   a. At the front of the list
   b. After a given node
   c. At the end of the list.

3. Write a program to ***delete a node*** from the linked list. A node can be deleted from the linked list using three ways: [Write code for all the three ways.]
   a. Delete from the beginning
   b. Delete from the end
   c. Delete from the middle.

4. Implement the ***circular linked list*** and perform the operation of traversal on it. In a conventional linked list, we traverse the list from the head node and stop the traversal when we reach NULL. In a circular linked list, we stop traversal when we reach the first node again.

5. Implement the ***doubly linked list*** and perform the deletion and/ or insertion operation on it. Again, you can perform insertion deletion according to the three ways as given above. Implement all of them according to availability of time.

Experiment No. 5 [Stack Application]

1. Implement a stack using an array and using a linked list.

2. Given a stack, sort it using recursion. Use of any loop constructs like `while`, `for`, etc. is not allowed. We can only use the following functions on Stack S:

a.	isEmpty(S):	Tests whether stack is empty or not.
b.	push(S):	Adds new element to the stack.
c.	pop(S):	Removes top element from the stack.
d.	top(S):	Returns value of the top element.

Experiment No. 6 [Stack Applications]

1. Convert the given infix expression into postfix expression using stack.
   Example- Input: $a + b * (c\^{}d - e)\^{}(f + g * h) - i$
   Output: $abcd\^{}e - fgh * +\^{} * +i -$

2. Write a program to evaluate the following given postfix expressions:

a.	$2\ 3\ 1\ *\ +9\ -$	Output: -4
b.	$2\ 2\ +\ 2\ /\ 5\ *\ 7\ +$	Output: 17

3. Given an expression, write a program to examine whether the pairs and the orders of "{", "}", "(", ")", "[", "]" are correct in the expression or not.
   Example: Input: exp = "[( )]{ }{[( )( )]( )}" Output: Balanced
   Input: exp = "[( ])" Output: Not Balanced

Practical No. 7 [Queue]

1. Implement various functionalities of Queue using Arrays. For example: insertion, deletion, front element, rear element etc.

2. Implement various functionalities of Queue using Linked Lists. Again, you can implement operation given above.

3. Implement Priority Queue, where every element has a priority associated with it. Perform operations like Insertion and Deletion in a priority queue.

4. Implement Double Ended Queue that supports following operation:
   a. insertFront(): Adds an item at the front of Deque.
   b. insertLast(): Adds an item at the rear of Deque.
   c. deleteFront(): Deletes an item from the front of Deque.
   d. deleteLast(): Deletes an item from the rear of Deque.

5. Implement Double Ended Queue that supports following operation:
   a. getFront(): Gets the front item from the queue.
   b. getRear(): Gets the last item from queue.
   c. isEmpty(): Checks whether Deque is empty or not.
   d. isFull(): Checks whether Deque is full or not.

Practical No. 8 [Trees]

1. Write a program to insert an element, delete an element and search an element in the Binary Tree.

2. Study and implement the Binary Tree and perform following three types of traversals in a binary tree:
   a. Pre-order Traversal
   b. In-order Traversal
   c. Post-order Traversal

3. Given a preorder traversal sequence of a Binary Search Tree, construct the corresponding Binary Search Tree.

Practical No. 9 [Graphs]

1. For a given graph $G = (V, E)$, study and implement the Breadth First Search (or traversal) i.e., BFS. Also, perform complexity analysis of this algorithm in-terms of time and space.

2. For a given graph $G = (V, E)$, study and implement the Depth First Search (or traversal) i.e., DFS. Also, perform complexity analysis of this algorithm in-terms of time and space.

3. Given a directed graph, check whether the graph contains a cycle or not. Your function should return true if the given graph contains at least one cycle, else return false. Perform same task for undirected graph as well.

4. Implement Minimum Spanning Tree (MST) using the greedy Kruskal's algorithm. A MST or minimum weight spanning tree for a weighted, connected, undirected graph is a spanning tree with a weight less than or equal to the weight of every other spanning tree. The weight of a spanning tree is the sum of weights given to each edge of the spanning tree.

Practical No. 10 [Hashing]

1. Given a limited range array containing both positive and non-positive numbers, i.e., elements are in the range from -MAX to +MAX. Our task is to search if some number is present in the array or not in O(1) time.

2. Given two arrays: *A* and *B*. Find whether *B* is a subset of *A* or not using Hashing. Both the arrays are not in sorted order. It may be assumed that elements in both arrays are distinct.

Practical No. 11, 12, 13 [Mini Project and Problem Solving]

These practicals are dedicated for mini projects and doubt solving sessions. Students are expected to do a mini project by forming a group of 5 on the concepts they have learned in data structures. They are free to select any mini project as per their choice. If they are not able to decide, they can select any one from the following:

1. **Mini Voting System:**
   An online voting system is a software platform that enables organizations to conduct votes and elections securely. A high-quality online voting system strikes a balance between ballot security, convenience, and the overall needs of a voting event. By collecting the input of your group in a systematic and verifiable manner, online voting tools and online election voting systems assist you in making crucial decisions. These decisions are frequently taken on a yearly basis – either during an event (such as your organization's AGM) or at a specific time of the year. Alternatively, you may conduct regular polls among your colleagues (e.g., anonymous student feedback surveys).

   With this voting system, users can enter their preferences and the total votes and leading candidate can be calculated. It's a straightforward Data structure project that's simple to grasp. Small-scale election efforts can benefit from this.

2. **Library Management System:**
   Library management is a project that manages and preserves electronic book data based on the demands of students. Both students and library administrators can use the system

to keep track of all the books available in the library. It allows both the administrator and the student to look for the desired book. You can include the functionalities like searching book, issue books, view available books and more using various data structures.

3. **Electricity Bill Calculator:**
   The Electricity Cost Calculator project is an application-based micro project that predicts the following month's electricity bill based on the appliances or loads used. Visual studio code was used to write the code for this project. This project employs a multi-file and multi-platform strategy (Linux and Windows). People who do not have a technical understanding of calculating power bills can use this program to forecast their electricity bills for the coming months; however, an electricity bill calculator must have the following features:
   A. All loads' power rating
   B. Unit consumed per day
   C. Units consumed per month, and
   D. Total load calculation

4. **Movie Ticket Booking System:**
   The project's goal is to inform a consumer about the MOVIE TICKET BOOKING SYSTEM so that they can order tickets. The project should have the goal of making the process as simple and quick as possible. The user can book tickets, cancel tickets, and view all booking records using the system. This project's major purpose is to supply various forms of client facilities as well as excellent customer service. It should meet nearly all the conditions for reserving a ticket.

5. **Bus Reservation System:**
   This mini project is built on the concept of booking bus tickets in advance. The user can check the bus schedule, book tickets, cancel reservations, and check the bus status board using this system. When purchasing tickets, the user must first enter the bus number, after which the system will display the entire number of bus seats along with the passengers' names, and the user must then enter the number of tickets, seat number, and person's name.
   This project will require use of arrays, if-else logic, loop statements, and various functions like login( ), cancel( ), etc. for its implementation.