CONTENTS

Program 1

Title: To perform Insertion, deletion and display operations on an array of structures.

Aim: To perform the above operations by implementing a menu and learn the mechanism by which structure variables are created and handled.

Input: array of structures with the structure members being of integer, float and string data types.

Expected result: To create an array of structures with its members suitably populated by the user through a function, suitably deleted and displayed through respective functions.

Outcome of the task:

- ***Learn to define a structure with its members.***
- ***Learn to declare and define functions with the structure variable as an argument.***
- ***Learn to create a menu with the necessary programming constructs.***
- ***Learn to write the insert, delete and display operations on an array of structures.***

Program 2

Title: To perform basic sorting techniques like bubble, selection and insertion sorts on an array of structures.

Aim: To perform the above sorting operations by implementing a menu with the sorts as options and learn the mechanism by which structure variables are sorted accordingly.

Input: array of structures with the structure members being of integer, float and string data types.

Expected result: To create an array of structures with its members suitably populated by the user and to perform the various sorts on the created structure variables suitably.

Outcome of the task:

- ***Learn the above sorting techniques.***
- ***Learn to apply it accordingly on the array of structures.***
- ***Learn to display the sorted structure details in a suitable tabular fashion with all fields.***

Program 3

Title: To perform advanced sorting techniques like quick and merge sorts on an array of structures.

Aim: To perform the above sorting operations by implementing a menu with the sorts as options and learn the mechanism by which structure variables are sorted accordingly.

Input: array of structures with the structure members being of integer, float and string data types.

Expected result: To create an array of structures with its members suitably populated by the user and to perform the various sorts on the created structure variables suitably.

Outcome of the task:

- ***Learn the above sorting techniques.***
- ***Learn to apply it accordingly on the array of structures.***
- ***Learn to display the sorted structure details in a suitable tabular fashion with all fields.***

Program 4

Title: To implement singly linked lists

Aim: Learn to create a singly linked list and perform insertion, deletion, display and search operations on it by implementing a menu with the operations as options.

Input: Singly linked list having a structure with suitably defined members such as

1. Integer/char
2. Pointer to the structure.

Expected result: To create a singly linked list with a suitably defined structure. Also to be able to insert a node, delete a node and display the nodes of the list suitably.

Outcome of the task:

- ***Learn to create a singly linked list***

- ***Learn the use of -> * operators and dynamic memory allocation functions.***
- ***Learn inserting a node into the list (at the front, rear and any arbitrary location).***
- ***Also learn deleting a node from the list (at specified position for e.g.).***
- ***Display the created singly linked list and show the effect of the various operations suitably.***

.

Program 5

Title: To implement circular linked lists

Aim: Learn to create a circular linked list and perform insertion, deletion, display and search operations on it by implementing a menu with the operations as options.

Input: Circular linked list having a structure suitably defined members such as

1. Integer/char
2. Pointer to the structure.

   Also create the list a) with Header node b) w/o header node

Expected result: To create a circular linked list with a suitably defined structure. Also to be able to insert a node, delete a node and display the nodes of the list suitably.

Outcome of the task:

- ***Learn to create a circular linked list***
- ***Learn inserting a node into the list (at the front, rear and any arbitrary location).***
- ***Also learn deleting a node from the list (at specified position for e.g.).***
- ***Display the created circular linked list and show the effect of the various operations suitably with/without Header node.***

Program 6

Title: To implement doubly linked lists

Aim: Learn to create a doubly linked list and perform insertion, deletion, display and search operations on it by implementing a menu with the operations as options.

Input: Doubly linked list having a structure suitably defined members such as
1. Integer/char
2. Pointer to the structure which points to the next node
3. Pointer to the structure which points to the previous node.
Also create the list a) with Header node b) w/o header node

Expected result: To create a doubly linked list with a suitably defined structure. Also to be able to insert a node, delete a node and display the nodes of the list suitably.

Outcome of the task:

- ***Learn to create a doubly linked list.***
- ***Learn inserting a node into the list (at the front, rear and any arbitrary location).***
- ***Also learn deleting a node from the list (at specified position for e.g.).***
- ***Display the created doubly linked list and show the effect of the various operations suitably with/without Header node.***

Program 7

Title: To implement stacks using arrays
Aim: Learn to implement a stack and perform push, pop and display operations on it by implementing a menu with the operations as options.

Input: Array of user defined size with the array having elements of either integer/char data types.

Expected result: To create a stack with a suitably defined MAX size. Also to be able to push/pop an element and display the contents of the stack with the stack TOP.

Outcome of the task:

- ***Learn to create a stack using arrays.***
- ***Implement push operation with Stack overflow condition.***
- ***Implement pop operation with stack empty condition.***
- ***Display the created stack with stack top***
- ***Understand why a stack is a LIFO data structure.***

Program 8

Title: Application of stacks to convert expressions

Aim: Learn to implement a stack for converting expressions from one form to another using the Push and Pop operations on it. The different operations are infix to postfix, infix to prefix, postfix to prefix

Input: 1. Array of user defined size with the array having elements of either integer/char data types.
2. Expression of desired form.

Expected result: To create a stack with a suitably defined MAX size. Also to be able to push/pop an element and display the contents of the stack with the stack TOP. Perform the various conversion operations by implementing a menu with the operations as options

Outcome of the task:

- ***Learn to create a stack using arrays with stack overflow and stack empty conditions.***
- ***Display the created stack with stack top.***
- ***Learn to apply the stack data structure to convert expressions.***
- ***Learn about input precedence and stack precedence when comparing characters and perform push and pop operations suitably.***

Program 9

Title: Application of stacks to evaluate expressions

Aim: Learn to implement a stack for evaluating expressions using the Push and Pop operations on it. The expressions to be evaluated are postfix and prefix.

Input 1. Array of user defined size with the array having elements of either integer/char data types.
2. Expression of desired form.

Expected result: To create a stack with a suitably defined MAX size. Also to be able to push/pop an element and display the contents of the stack with the stack TOP. Perform the various evaluation operations by implementing a menu with the operations as options

Outcome of the task:

- ***Learn to create a stack using arrays with stack overflow and stack empty conditions.***
- ***Display the created stack with stack top.***
- ***Learn to apply the stack data structure to evaluate expressions.***
- ***Learn about input precedence and stack precedence when comparing characters and perform push and pop operations suitably.***

Program 10

Title: To implement stacks using linked lists

Aim: Learn to implement a stack using linked lists and perform Push, pop and display operations on it by implementing a menu with the operations as options.

Input: Doubly linked list having a structure with suitably defined members such as
1. Integer/char
2. Pointer to the structure which points to the next node
3. Pointer to the structure which points to the previous node.

Expected result: To create a stack with a suitably defined MAX size. Also to be able to push/pop an element and display the contents of the stack with the stack TOP.

Outcome of the task:

- ***Learn to create a stack using linked lists.***
- ***Implement push operation with Stack overflow condition.***

- ***Implement pop operation with stack empty condition.***
- ***Display the created stack with stack top***
- ***Understand why a stack is a LIFO data structure.***

Program 11

Title: To check if a string/number is a palindrome using stacks

Aim: Learn to check if a string/ number are a palindrome using stacks by performing Push and Pop operations on it.

Input: 1. Array of user defined size with the array having elements of either integer/char data types.
2. User input string/number.

Expected result: To create a stack with a suitably defined MAX size. Also to be able to push/pop an element and display if the user input string/number is a palindrome or not.

Outcome of the task:

- ***Learn to create a stack using arrays with stack overflow and stack empty conditions.***
- ***Display the created stack with stack top.***
- ***Understand to use a stack to check for palindrome***
- ***Learn about input precedence and stack precedence when comparing characters and perform push and pop operations suitably.***

Program 12

Title: To implement queues using arrays

Aim: Learn to implement a queue and perform enqueue, dequeue and display operations on it by implementing a menu with the operations as options.

Input: Array of user defined size with the array having elements of either integer/char data types.

Expected result: To create a queue with a suitably defined QUEUE size. Also be able to insert/delete an element and display the contents of the queue with the front and rear respectively.

Outcome of the task:

- ***Learn to create a queue using arrays.***
- ***Implement enqueue operation at the queue rear with Queue full condition.***
- ***Implement dequeue operation at the queue front with Queue empty condition.***
- ***Display the created queue with queue front and queue rear.***
- ***Understand why a queue is a FIFO data structure.***

Program 13

Title: To implement circular queues using linked lists

Aim: Learn to implement a circular queue and perform enqueue, dequeue and display operations on it by implementing a menu with the operations as options.

Input: doubly linked list of suitable user defined size having elements of either integer/char data types.

Expected result: To create a circular queue with a suitably defined QUEUE size. Also be able to insert/delete an element and display the contents of the circular queue with the front and rear respectively.

Outcome of the task:

- ***Learn to create a circular queue using linked lists.***
- ***Implement enqueue operation at the queue rear with Circular Queue full condition.***
- ***Implement dequeue operation at the queue front with Circular Queue empty condition.***
- ***Display the created queue with queue front and queue rear.***

Program 14

Title: To implement priority queues using linked lists

Aim: Learn to implement a priority queue and perform enqueue, dequeue and display operations on it by implementing a menu with the operations as options.

Input: doubly linked list of suitable user defined size having elements of both integer/char data types and an additional integer field called the priority.

Expected result: To create a priority queue with a suitably defined QUEUE size. Also be able to insert/delete an element based on the priority and display the contents of the priority queue with the front and rear respectively.

Outcome of the task:

- ***Learn to create a priority queue using linked lists.***
- ***Implement enqueue operation at the queue rear with Circular Queue full condition.***
- ***Implement dequeue operation at the queue front with Circular Queue empty condition.***
- ***Display the created priority queue with queue front and queue rear.***

Program 15

Title: To implement double ended queues using linked lists

Aim: Learn to implement a double ended queue and perform enqueue, dequeue and display operations on it by implementing a menu with the operations as options.

Input: doubly linked list of suitable user defined size having elements of either integer/char data types.

Expected result: To create a double ended queue with a suitably defined QUEUE size. Also be able to insert/delete an element and display the contents of the double ended queue with the front and rear respectively.

Outcome of the task:

- ***Learn to create a double ended queue using linked lists.***
- ***Implement enqueue operation at the queue front/rear with Double ended queue full condition.***
- ***Implement dequeue operation at the queue front/rear with double ended queue empty condition.***
- ***Display the created priority queue with queue front and queue rear.***

Program 16

Title: To implement binary trees using linked lists

Aim: Learn to implement binary trees using linked lists and perform operations on it like inserting a node, deleting a node and displaying the contents of the tree by various traversal methods. A menu with all the operations is given as options. Various **traversals techniques** are **Preorder, Inorder** and **Postorder**

Input: doubly linked list having elements of integer data types with the two structure pointers pointing to the left and right child.

Expected result: To create a binary tree using doubly linked lists. Also be able to insert/delete a node from and display the contents of the binary tree with the root node respectively.

Outcome of the task:

- ***Learn to create a binary tree using doubly linked lists.***
- ***Insert left / child to a parent node in a proper fashion.***
- ***Delete left/right child to a parent node in a proper fashion.***
- ***Display the created binary tree suitably using any of the three traversal methods.***
- ***Learn the non-recursive method of implementing the traversals.***

Program 17

Title: To implement binary search trees (BST) using linked lists

Aim: Learn to implement binary search trees (BST) using linked lists and perform operations on it like inserting a node, deleting a node and displaying the contents of the tree by various traversal methods. A menu with all the operations is given as options.

Input: doubly linked list having elements of integer data type with the two structure pointers pointing to the left and right child

Expected result: To create a binary search tree using doubly linked lists where the left child is lesser than the parent which in turn is lesser than the right child. Also be able to insert/delete a node from and display the contents of the binary tree with the root node respectively.

Outcome of the task:

- ***Learn to create a binary search tree using doubly linked lists.***
- ***Insert left / child to a parent node in a proper fashion checking the value of the parent node.***
- ***Delete left/right child to a parent node in a proper fashion checking the value of the parent node and reconstruct the new binary tree suitably.***
- ***Display the created binary search tree suitably using any of the three traversal methods.***
- ***Learn the recursive method of implementing the traversals.***

Program 18

Title: To create the min heap and max heap

Aim: Learn to create min heap and max heap using doubly linked lists and perform operations on it like inserting a node, deleting a node and displaying the contents of the tree by various traversal methods. A menu with all the operations is given as options. Various **traversals techniques** are **Preorder, Inorder** and **Postorder**

Input: doubly linked list having elements of integer data types with the two structure pointers pointing to the left and right child.

Expected result: To create a min heap and max heap using doubly linked lists. Also be able to insert/delete a node from and display the contents of the heap tree with the root node respectively.

Outcome of the task:

- ***Learn to create a binary tree using doubly linked lists.***
- ***Learn to create min heap and max heap from a given set of user inputs***
- ***Display the min and max heaps suitably using any of the three traversal methods.***

Program 19

Title: To implement breadth first search in a given graph

Aim: Learn to search for a key element in a graph created using matrices.

Input: matrix having elements of integer data types and a queue to store intermediate results.

Expected result: To search for a key element in the tree and display "Search Success" in case of presence or "Search Failure" otherwise. Also display the path to the search of the key in both cases

Outcome of the task:

- ***Learn to represent a binary tree using arrays.***
- ***Learn to dequeue the queue which has all the elements of the tree properly in the search of the key element***
- ***Display the appropriate message in case of key element search and display the path to the***

search of the key in both cases.
- ***Learn the difference between DFS and BFS***

Program 20

Title: To implement depth first search in a given graph

Aim: Learn to search for a key element in a graph created using matrices.

Input: matrix having elements of integer data types and a queue to store intermediate results.

Expected result: To search for a key element in the tree and display “Search Success” in case of presence or “Search Failure” otherwise. Also display the path to the search of the key in both cases

Outcome of the task:

- ***Learn to represent a binary tree using arrays.***
- ***Learn to dequeue the queue which has all the elements of the tree properly in the search of the key element***
- ***Display the appropriate message in case of key element search and display the path to the search of the key in both cases.***
- ***Learn the difference between DFS and BFS***