**Pak-Austria Fachhochschule: Institute of Applied Sciences & Technology, Haripur, Pakistan**

**School of Computing Science**



**DSA**

**Submitted By: Haider Nasir**

**Registration # B24S0308AI088**

**Submitted to: Mr. Aamir Malik**

**Date of submission: 04/26/2025**

**Pak-Austria Fachhochschule: Institute of Applied Sciences & Technology, Haripur, Pakistan**

**School of Computing Science**

ALGORITHMS:

1. Insert at the beginning:

* Create a new node.
* Set its next to the current head.
* Make the new node the head.

2. Insert at the end:

* Create a new node.
* Traverse to the last node.
* Set the last node's next to the new node.

3. Insert after a given node:

* Create a new node.
* Set new node's next to the given node's next.
* Update given node's next to the new node.

4. Delete a node by value:

* Traverse the list to find the node before the target.
* Update its next to skip the target node.
* Delete target node.

5. Search for a value:

* Traverse and compare each node's data.
* If found, return position; otherwise, not found.

6. Display the list:

* Traverse from head to end printing each node.

Question 1B: Apply above operations for Double and Circular Linked Lists similarly

Differences:

* Double: Uses prev pointers too.
* Circular: Last node points back to head.

***CPP FILES ARE LINKED WITH IT HAVING CODE OF ALL three***

Question 2: Expression Conversions Infix: (A - (B + C)) $ D $ (E + F) \* G / H A. Infix to Postfix:

Infix: (A - (B + C)) $ D $ (E + F) \* G / H

Postfix: A B C + - D $ E F + G \* H / $

Postfix to Infix:

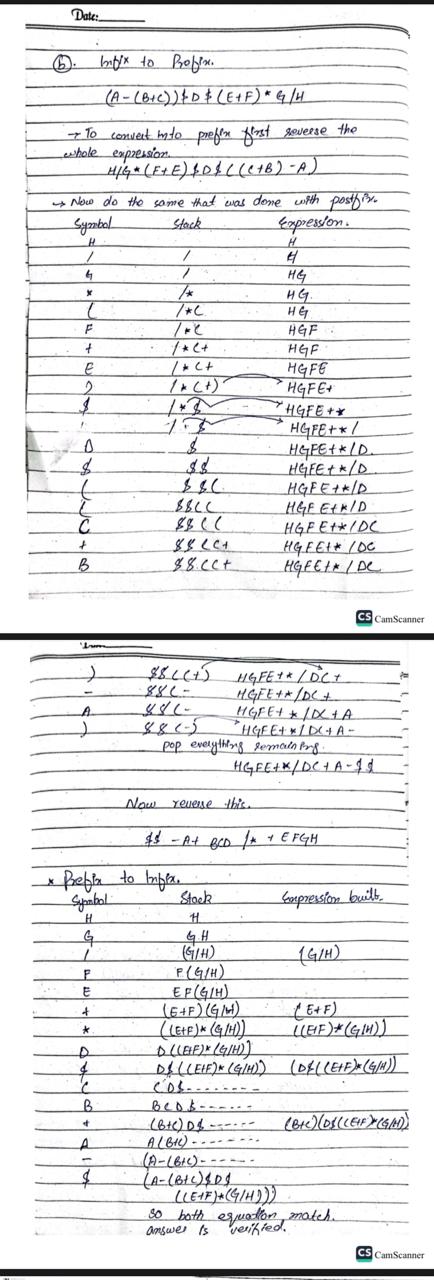
=> ((A - (B + C)) $ D) $ ((E + F) \* G / H)

B. Infix to Prefix:

Prefix: $ $ - A + B C D / \* + E F G H

Prefix to Infix:

=> ((A - (B + C)) $ D) $ ((E + F) \* G / H)



Question 3: Queue Implementation (Logic Only)

* Enqueue: Add at rear.
* Dequeue: Remove from front.
* Peek: Show front.
* IsEmpty: Check if front == NULL.
* IsFull: (For array) check rear == size-1.
* Traverse: From front to rear.

Example: Queue = [10, 20, 30]

Enqueue(40) => [10, 20, 30, 40]

Dequeue() => [20, 30, 40]

Peek => 20

Question 4: Tree Traversals (Binary Search Tree)

Given BST Example: Insert nodes in order: 50, 30, 70, 20, 40, 60, 80

Preorder: 50, 30, 20, 40, 70, 60, 80

Inorder: 20, 30, 40, 50, 60, 70, 80

Postorder: 20, 40, 30, 60, 80, 70, 50

Level Order: 50, 30, 70, 20, 40, 60, 80