LAB SESSION 4: LINKED LIST IMPLEMENTATION OF STACK AND QUEUE

AIM: To implement Stack and Queue using Linked list

PROBLEM DEFINITION:

1. Develop a C program to implement Queues using Linked List
2. Develop a C program to implement Stacks using Linked List and use the same to convert a postfix expression to its equivalent infix expression.

THEORY:

In Stack implementation using linked-list, the nodes are maintained in non-contiguous memory. Each node contains a pointer to the immediate next in line node in the Stack.

Adding a new node in the Stack is termed a push operation.
Push operation on stack implementation using linked-list involves several steps:

- Create a node first and allocate memory to it.
- If the list is empty, then the node is pushed as the first node of the linked list. This operation assigns a value to the data part of the node and gives NULL to the address part of the node.
- If some nodes are already in the linked list, then we have to add a new node at the beginning to the list not to violate the Stack's property. For this, assign the element to the address field of the new node and make a new node which will be starting node of the list.
- An overflow condition occurs when we try to push an operation if the Stack is already full.

Deleting a node from the Stack is known as a pop operation.
To perform the pop operation involves the following steps:

- In Stack, the node is removed from the end of the linked list. Therefore, must delete the value stored in the head pointer, and the node must get free. The following link node will become the head node now.
- An underflow condition will occur when we try to pop an operation when the Stack is already empty. The Stack will be meaningless if the head pointer of the list points to NULL.

In Queue implementation using linked-list, to keep track of the front and rear node, two pointers are preserved in the memory. The first pointer stores the location where the queue starts, and another pointer keeps track of the last data element of a queue.

The insert operation appends the queue by adding an element to the end of the queue. The new element will be the last element of the queue.

- Firstly, allocate the memory for the new node ptr
- There can be the two scenario of inserting this new node ptr into the linked queue.

  In the first scenario, we insert element into an empty queue. Now, the new element will be added as the only element of the queue and the next pointer of front and rear pointer both, will point to NULL.

  In the second case, the queue contains more than one element. In this scenario, we need to update the end pointer rear so that the next pointer of rear will point to the new node ptr. We also need to make the rear pointer point to the newly added node ptr. We also need to make the next pointer of rear point to NULL.

Deletion operation removes the element that is first inserted among all the queue elements.

- Firstly, we need to check either the list is empty or not.

  The condition front == NULL becomes true if the list is empty, in this case , we simply write underflow on the console.

  Otherwise, we will delete the element that is pointed by the pointer front. For this purpose, copy the node pointed by the front pointer into the pointer ptr. Now, shift the front pointer, point to its next node and free the node pointed by the node ptr.

ALGORITHM

1. Convert a postfix expression to its equivalent infix expression.

PROGRAM AND OUTPUT:

CONCLUSION: