Lab 8: Implementation of Stack and Linear Queue using singly linked lists.

Theory:

Stacks can be implemented using both arrays and linked lists. Linked list implementation of stack is also called dynamic implementation of stack.

Structure for Linked List Implementation of Stack:

```
struct st
{
        int info;
        st *next;
}
typedef struct st NodeType;
NodeType *top;
```

Linked List Implementation of Stack – PUSH:

1. Create a NewNode with a given value.
2. If(top==NULL)
   Set, NewNode→next=NULL;
3. Else, Set, NewNode→next=top
   Set, top=NewNode
4. End

Linked List Implementation of Stack – POP:

1. Start
2. If(top==NULL)

Display "Stack Underflow" and terminate

3. Else, temp=top;
   top=top→next;
4. free(temp)
5. Stop

Similarly, Linear Queues can also be implemented using Singly Linked Lists.

Structure for Linked List Implementation of Linear Queue:

```
struct node
{
   int info;
   struct node *next;
};
typedef struct node NodeType;
NodeType *front,*rear;
```

Linked List Implementation of Linear Queue – ENQUEUE:

```
if(front==NULL)
   {
      front=rear=NewNode;
   }
  else if(front->next==NULL)
   {
      front->next=NewNode;
      rear=NewNode;
   }
   else
   {
      rear->next=NewNode;
      rear=NewNode;
   }
```

Linked List Implementation of Linear Queue – DEQUEUE:

```
if(front==NULL)
   {
      Display "QUEUE is EMPTY";
      return;
   }
   else if(front->next==NULL)
   {
      temp=front;
      front=rear=NULL;
     free(temp);
   }
   else
   {
      temp=front;
      front=front->next;
      free(temp);
   }
```

Source Code for Linked List Implementation of Stack :

```
#include <stdio.h>
#include <stdlib.h>
struct SLL
{
    int info;
    struct SLL *next;
};
typedef struct SLL NodeType;
NodeType *top;
void push(NodeType *S,int element);
int pop(NodeType *S);
int peek(NodeType *S);
int main()
{
    int choice,element;
    NodeType Stack;
    top = NULL;
    do
    {
        printf("1.PUSH\n2.POP\n3.PEEK\n4.EXIT\n");
        printf("Your choice? ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
                printf("Enter element : ");
                scanf("%d",&element);
                push(&Stack,element);
                break;

            case 2:
                if(top == NULL)
                    printf("STACK UNDERFLOW\n");
                else
                {
```

```
                    printf("%d WAS POPPED\n",pop(&Stack));
                }
                break;

            case 3:
                if(top == NULL)
                    printf("STACK IS EMPTY\n");
                else
                {
                    printf("%d IS AT TOP\n",peek(&Stack));
                }
                break;

            case 4:
                printf("BYE\n");
        }
    } while (choice != 4);
    return 0;
}

void push(NodeType *S,int element)
{
    NodeType *NewNode;
    NewNode = (NodeType*)malloc(sizeof(NodeType));
    if(NewNode == NULL)
        printf("MEMORY ALLOCATION FAILED\n");
    else
    {
        NewNode->info = element;
        NewNode->next = NULL;
        //Case 1: Stack is Empty i.e top == NULL
        if(top == NULL)
            top = NewNode;
        else
        {
        //Case 2: Stack is not empty
```

```
            NewNode->next = top;
            top = NewNode;
        }
        printf("%d was successfully PUSHED\n",top->info);
    }
}
int pop(NodeType *S)
{
    NodeType *temp;
    int ret= top->info;
    //Case 1: When there is only a single element on stack
    if(top->next == NULL)
    {
        temp = top;
        free(temp);
        top = NULL;
    }
    else
    {
    //Case 2: When there are multiple elements in the stack
        temp = top;
        top = top->next;
        free(temp);
    }
    return ret;
}
int peek(NodeType *S)
{
    return top->info;
}
```

Source Code for Linked List Implementation of Linear Queue :

```
#include <stdio.h>
#include <stdlib.h>
struct SLL
{
    int info;
    struct SLL *next;
};
typedef struct SLL NodeType;
NodeType *front,*rear;
void enqueue(NodeType *S,int element);
int dequeue(NodeType *S);
int checkF(NodeType *S);
int checkR(NodeType *S);
int main()
{
    int choice,element;
    NodeType Queue;
    front = NULL;
    rear = NULL;
    do
    {
        printf("1.ENQUEUE\n2.DEQUEUE\n3.CHECK F");
        printf("\n4.CHECK R\n5.EXIT\n");
        printf("Your choice? ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
                printf("Enter element : ");
                scanf("%d",&element);
                enqueue(&Queue,element);
                break;

            case 2:
                if(front == NULL)
```

```
                printf("QUEUE IS EMPTY\n");
            else
            {
                printf("%d WAS DEQUEUED\n",dequeue(&Queue));
            }
            break;

        case 3:
            if(front == NULL)
                printf("QUEUE IS EMPTY\n");
            else
            {
                printf("%d IS AT FRONT\n",checkF(&Queue));
            }
            break;

        case 4:
            if(front == NULL)
                printf("QUEUE IS EMPTY\n");
            else
            {
                printf("%d IS AT REAR\n",checkR(&Queue));
            }
            break;

        case 5:
            printf("BYE\n");
    }
    } while (choice != 5);
    return 0;
}

void enqueue(NodeType *S,int element)
{
    NodeType *NewNode;
    NewNode = (NodeType*)malloc(sizeof(NodeType));
```

```
    if(NewNode == NULL)
        printf("MEMORY ALLOCATION FAILED\n");
    else
    {
        NewNode->info = element;
        NewNode->next = NULL;
        //Case 1: Stack is Empty i.e top == NULL
        if(front == NULL)
        {
            front = NewNode;
            rear = NewNode;
        }
        else
        {
        //Case 2: Stack is not empty
            rear->next = NewNode;
            rear = NewNode;
        }
        printf("%d was successfully ENQUEUED\n",rear->info);
    }
}
int dequeue(NodeType *S)
{
    NodeType *temp;
    int ret= front->info;
    //Case 1: When there is only a single element on queue
    if(front->next == NULL)
    {
        temp = front;
        free(temp);
        front = NULL;
        rear = NULL;
    }
    else
    {
    //Case 2: When there are multiple elements in the queue
```

```
        temp = front;
        front = front->next;
        free(temp);
    }
    return ret;
}
int checkF(NodeType *S)
{
    return front->info;
}

int checkR(NodeType *S)
{
    return rear->info;
}
```