EXPERIMENT - 4

Data Structures

Aim

Implementation of stack using linked list.

Syeda Reeha Quasar
14114902719
3C7

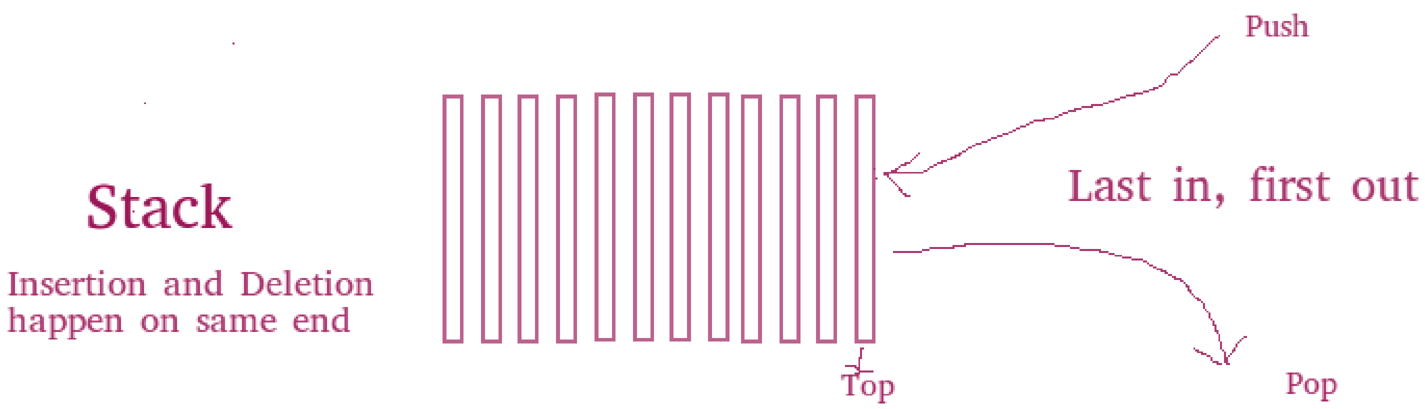
Name - Syeda Reeha Quasar **Roll no. – 14114802719** **Group – C7**

EXPERIMENT – 4

AIM: Implementation of stack using linked list.

THEORY

Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO(Last In First Out) or FILO(First In Last Out).



There are many real-life examples of a stack. Consider an example of plates stacked over one another in the canteen. The plate which is at the top is the first one to be removed, i.e. the plate which has been placed at the bottommost position remains in the stack for the longest period of time. So, it can be simply seen to follow LIFO(Last In First Out)/FILO(First In Last Out) order.

A linked list is a linear data structure where each element is a separate object. Linked list elements are not stored at contiguous location; the elements are linked using pointers. Each node of a list is made up of two items - the data and a reference to the next node. The last node has a reference to null.

Displaying the contents of a linked list is very simple. We keep moving the temp node to the next one and display its contents.

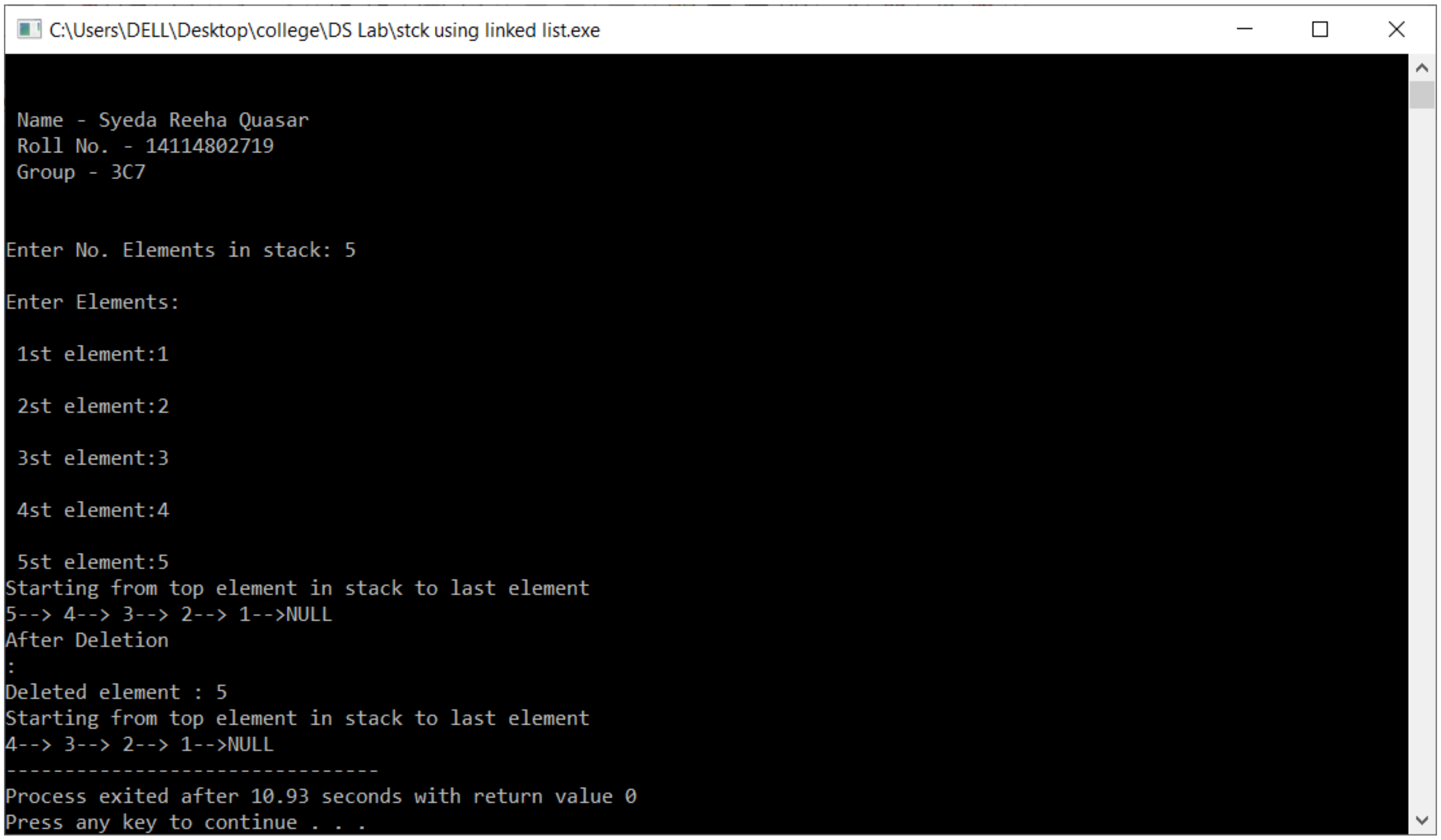
Source code:

```
//stack implementation using linked list

#include<stdio.h>
#include<conio.h>
#include<stdlib.h>

//node structure
struct node{
        int data;
        struct node *next;
}

*top = NULL;

//push for stack
void push (int x){
        struct node*t;
        t = (struct node*)malloc(sizeof(struct node));
        t->data=x;
   if (top == NULL) {
        t->next=NULL;
        }
        else{
                t->next=top;
        }
        top=t;
}
```

```
// pop for stack
void pop(){
        if (top == NULL) {
                printf("\nStack is Empty\n");
        }
        else {
                struct node*temp=top;
                printf("\nDeleted element : %d\n",temp->data);
                top=top->next;
                free(temp);
        }
}

//function for printing stack elements
void display(){
        if (top == NULL) {
                printf("\nStack is  Empty\n");
        }
        else {
                struct node*t;
                t=top;
                printf("Starting from top element in stack to last element \n");
                while(t->next!=NULL) {
                        printf("%d--> ",t->data);
                        t=t->next;
                }
                printf("%d-->NULL",t->data);
        }
}
```

```
int main(){

	// my info

	printf("\n\n Name - Syeda Reeha Quasar \n Roll No. - 14114802719 \n Group - 3C7 \n\n");


	int n,x;

	// entering data

	printf("\nEnter No. Elements in stack: ");

	scanf("%d",&n);

	printf("\nEnter Elements:\n");


	//insertion

	for(int i=1;i<=n;i++){

		printf("\n %dst element:",i);

		scanf("%d",&x);

		push(x);

	}


	display(); // orignal stack


	//deletion

	printf("\nAfter Deletion\n:");

	pop();


	//display after deletion

	display();


	return 0;

}
```

OUTPUT



C:\Users\DELL\Desktop\college\DS Lab\stck using linked list.exe

```
 Name - Syeda Reeha Quasar
 Roll No. - 14114802719
 Group - 3C7


Enter No. Elements in stack: 5

Enter Elements:

 1st element:1

 2st element:2

 3st element:3

 4st element:4

 5st element:5
Starting from top element in stack to last element
5--> 4--> 3--> 2--> 1-->NULL
After Deletion
:
Deleted element : 5
Starting from top element in stack to last element
4--> 3--> 2--> 1-->NULL
--------------------------------
Process exited after 10.93 seconds with return value 0
Press any key to continue . . .
```

Name - Syeda Reeha Quasar **Roll no. – 14114802719** **Group – C7**

Viva Questions

Q1. The following postfix expression with single digit operands is evaluated using a stack: 8 2 3 ^ / 2 3 * + 5 1 * – Note that ^ is the exponentiation operator. The two elements of the stack after the first * is evaluated are:

a) 6, 1
b) 5, 7
c) 3, 2
d) 1, 5

Ans1.

a) 6, 1

8, 2, 3 gets pushed into stack then as ^ is found 2 are popped and evaluated => 2^3 = 8 then 8 Is pushed to stack then as we find / 8 and 8 are popped then 8/8 is 1 which is pushed then 2 and 3 are also pushed then we reach * so we do 2*3 = 6 which is pushed to stack. We have to stop here so stck has 6, 1 in it.

Q2. What is the best case time complexity of deleting a node in a singly linked list?

a) O (n)
b) O (n2)
c) O (n logn)
d) O (1)

Ans2.

a) O(1)

Deletion of the head node in the linked list is taken as the best case. The successor of the head node is changed to head and deletes the predecessor of the newly assigned head node. This process completes in O(1) time.

Q3. Which of the following statements are not correct with respect to Singly Linked List(SLL) and Doubly Linked List(DLL)?

a) **Complexity of Insertion and Deletion at known position is O(n) in SLL and O(1) in DLL**
b) **SLL uses lesser memory per node than DLL**
c) **DLL has more searching power than SLL**
d) **Number of node fields in SLL is more than DLL**

Ans3.

a) Number of node fields in SLL is more than DLL

To insert and delete at known positions requires complete traversal of the list in worst case in SLL, SLL consists of an item and a node field, while DLL has an item and two node fields, hence SLL occupies lesser memory, DLL can be traversed both ways(left and right), while SLL can traverse in only one direction, hence more searching power of DLL. Node fields in SLL is 2 (data and address of next node) whereas in DLL is 3(data, address to next node, address to previous node).

Q4. What does 'stack overflow' refer to?

a) **Accessing item from an undefined stack**
b) **Adding items to a full stack**
c) **Removing items from an empty stack**
d) **Index out of bounds exception**

Ans4.

b) **Adding items to a full stack**

A stack overflow occurs if the call stack pointer exceeds the stack bound. The call stack may consist of a limited amount of address space, often determined at the start of the program.

Q5. Consider these functions:

- **● push( ): push an element into the stack**
- **● pop( ): pop the top of the stack element**
- **● top( ): returns the storm stored in top of the stack node**

What will be the output after performing these sequences of operations?

- ➔ push(20);
- ➔ push(4);
- ➔ top( );
- ➔ pop( );
- ➔ pop( );
- ➔ pop( );
- ➔ push(5);
- ➔ top( );

a. 20
b. 4
c. Stack Underflow
d. 5

Ans-5.

a. 5