

DATA STRUCTURES AND ALGORITHMS

LAB 8

SUBMITTED BY:

Tanzeela Asghar

SUBMITTED TO:

SIR REHAN

REGISTRATION NO:

2021-BSE- 032

COURSE:

Data Structures

DEPARTMENT:

BSE-3A

Exercise 1

A stack can be implemented using a linked list. The first node can serve as the 'top' of Stack and 'push' and 'pop' operations can be implemented by adding and removing nodes at the head of the linked list. Implement the Stack class using a linked list and provide all the standard member functions.

PROGRAM

```
#include "stdafx.h"
#include<iostream>
using namespace std;
class list_STACK
{
    struct node
    {
        int data;
        node *next;
    }*top;

    public:
    void push(int x)
    {
        node *temp;
        temp=new node;
        temp->data=x;
        temp->next=NULL;
        if(top!=NULL)
            temp->next=top;
        top=temp;
    }
    int pop()
    {
        int q;
        node *temp;

        if(top==NULL)
        {
            cout<<"stack empty "<<endl;
            return -1;
        }

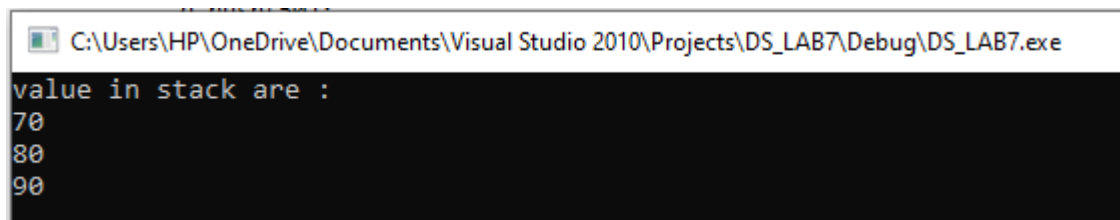
        temp=top;
        top=temp->next;
        q=temp->data;
        delete temp;
        return q;
    }
    void display()
    {
        node *temp;
        temp=top;
        cout<<"value in stack are : "<<endl;
        while(temp!=NULL)
```

```

        {
            cout<<temp->data<<endl;
            temp=temp->next;
        }
    };
int _tmain(int argc, _TCHAR* argv[])
{
    list_STACK q;
    q.push(90);
    q.push(80);
    q.push(70);
    q.push(60);
    q.push(50);
    q.pop();
    q.pop();
    q.display();
    system("pause");
    return 0;
}

```

OUTPUT



```

C:\Users\HP\OneDrive\Documents\Visual Studio 2010\Projects\DS_LAB7\Debug\DS_LAB7.exe
value in stack are :
70
80
90

```

Exercise 2

Implement simple Queue through linked list.

PROGRAM

```

#include "stdafx.h"
#include <iostream>
using namespace std;
class queue
{
private:
    struct node
    {
        int data;
        node *next;
    } *front , *rear;
public:
    queue ( )
    {
        front=NULL;
        rear=NULL;
    }
    void addq (int item)
    {
        node *temp;

```

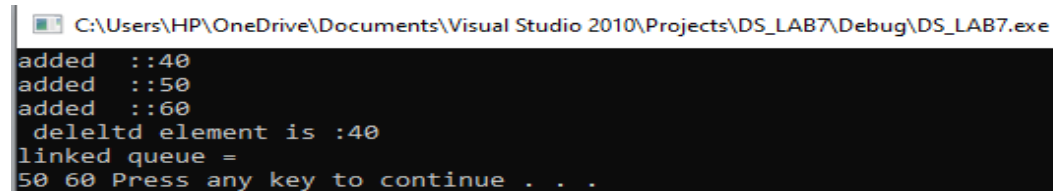
```

        temp=new node;
        temp->data=item;
        temp->next=NULL;
        if(front==NULL)
        {
            front=temp;
            rear=temp;
        }
        else
        {
            rear->next=temp;
            rear=temp;
        }

        cout<<"added  ::"<<item<<endl;
    }
    int delq ( )
    {
        node* temp;
        int item;
        temp=new node;
        temp=front;
        item=temp->data;
        front=front->next;
        delete temp;
        cout<<" deleted element is :";
        return item;
        cout<<endl;
    }
    void display ( )
    {
        node *temp;
        temp=front;
        while(temp!=NULL) {
            cout<<temp->data<<" ";
            temp=temp->next; }
    }
};
int _tmain(int argc, _TCHAR* argv[])
{
    queue q1;
    q1.addq(40);
    q1.addq(50);
    q1.addq(60);
    cout<<q1.delq()<<endl;
    cout<<"linked queue = "<<endl;
    q1.display();
    system("pause");
    return 0;
}

```

OUTPUT



```

C:\Users\HP\OneDrive\Documents\Visual Studio 2010\Projects\DS_LAB7\Debug\DS_LAB7.exe
added  ::40
added  ::50
added  ::60
deleted element is :40
linked queue =
50 60 Press any key to continue . . .

```

Exercise 3

Write the following C++ functions to realize the indicated functionality on a singly linked list of integers.

a. A function that prints only the even-numbered nodes of the list.

```
void print_even()
{
    node *temp;
    temp=top;
    while(top!=NULL)
    {
        cout<<temp->data<<endl;
        temp=temp->next->next;
    }
}
```

b. A function which takes two data values as arguments and swaps the nodes containing them.

```
void swap(int x,int y)
{
    int z;
    node *temp;
    temp=new node;
    node *temp1;
    temp1=new node;
    temp->data=x;

    temp1->data=y;
    cout<<"before swapping : "<<endl;
    cout<<temp->data<<endl;
    cout<<temp1->data<<endl;
    z=x;
    x=y;
    y=z;
    temp->data=x;
    temp1->data=y;
    cout<<"After swapping : "<<endl;
    cout<<temp->data<<endl;
    cout<<temp1->data<<endl;
}
```

c. A function that deletes the first half of the linked list nodes. You are required to first determine the total number of linked list nodes and then delete the first part of the linked list.

```
void half_list_deleted()
{
    node *temp=head;
    int count=0;
    int length;
```

```
while(temp!=NULL)
{
    count++;
    temp=temp->next;
}
length=count/2;
cout<<" deleted half list :"<<endl;
for(int i=1;i<=length; i++)
{
    temp=head;
    cout<<temp->data<<endl;
    head=head->next;
    delete temp;}

}
```