21) IMPLEMENTING THE STACKS USING THE DEQUEUE

This is the standard sum of stacks.

CODE OF THE PROGRAM:

```
#include<iostream>
using namespace std;

class node{
   public:
   int data;
   node *next=NULL;
   node*previous=NULL;
};

class Dequeue{
   private:
   node *head=NULL;
   node *tail=NULL;
   int size=0;

   public:
   void insert_first(int x){
      node *t=new node;
      t->data=x;
      if(head==NULL){
         head={tail=t};
         size++;
      }
      else{
         head->previous=t;
         t->next=head;
         head=t;
         size++;
      }
   }
   void insert_last(int x){
      node *t=new node;
      t->data=x;
      if(tail==NULL){
         head={tail=t};
         size++;
      }
      else{
         tail->next=t;
         t->previous=tail;
         tail=t;
         size++;
```

```
    }
}
int delete_first(){
    if(head==NULL){
        return -1;
    }
    else if(head==tail){
        int t=head->data;
        head={tail=NULL};
        size--;
        return t;
    }
    else{
        int t=head->data;
        node *temp=head;
        head=head->next;
        head->previous=NULL;
        delete(temp);
        size--;
        return t;
    }
}
int delete_last(){
    if(tail==NULL){
        return -1;
    }
    else if(head==tail){
        int t=head->data;
        node *temp=head;
        head={tail=NULL};
        delete(temp);
        size--;
        return t;
    }
    else{
        int t=tail->data;
        node *temp=tail;
        tail=tail->previous;
        tail->next=NULL;
        delete(temp);
        size--;
        return t;
    }
}
int get_last(){
    if(tail==NULL){
        return -1;
    }
    else{
        return tail->data;
```

```
        }
    }
    int get_first(){
        if(head==NULL){
            return -1;
        }
        else{
            return head->data;
        }
    }
    int get_size(){
        return size;
    }
};

class stack{
    private:
    Dequeue dq;
    public:
    void push(int x){
        dq.insert_last(x);
        return;
    }
    int pop(){
        return dq.delete_last();
    }
    int top(){
        return dq.get_last();
    }
    int size(){
        return dq.get_size();
    }
};

int main(){
    stack s;
    int n;
    cout<<"\n Enter the number of elements needed in the stack:";
    cin>>n;
    for(int i=0;i<n;i++){
        int t;
        cout<<"\n Enter the element:";
        cin>>t;
        s.push(t);
        cout<<"\n The stack top is "<<s.top()<<" and the size of the stack is "<<s.size();
    }
    for(int i=0;i<=n;i++){
        cout<<"\n Popping the element "<<s.pop()<<" the size of the stack is "<<s.size();
    }
    return 0;
```

```
}
```