**DSA PRACTICAL**

**NAME : Khushi Panwar**

**7 . WAP TO IMPLEMENT STACK USING ARRAYS**

#include<iostream>

#include<iomanip>

using namespace std;

class stacktype{

int top;

int arr[10];

public :

stacktype() {

top=(-1);

}

void push(int num);

int pop();

void display();

int isEmpty();

int isFull();

};

void stacktype::push(int num){

int ele;

for (int i=0; i<num; i++) {

cout<<"Enter the element : ";

cin>>ele;

arr[++top]=ele;

}

return;

}

int stacktype::pop(){

int ele=arr[top--];

return ele;

}

int stacktype::isEmpty(){

int flag;

if (top==-1) flag=1;

else flag=0;

return flag;

}

int stacktype::isFull(){

int flag;

if (top==sizeof(arr)-1) flag=1;

else flag=0;

return flag;

}

void stacktype::display(){

cout<<"-> DISPLAYING THE ARRAY : "<<endl<<endl;

for (int i=0; i<=top; i++)

{

cout<<setw(3)<<arr[i];

}

return;

}

void operationList(){

cout<<setw(20)<<"======================="<<endl;

cout<<"\t STACK OPERATIONS"<<endl;

cout<<"1. PUSH the element into stack"<<endl;

cout<<"2. POP the element from stack"<<endl;

cout<<"3. DISPLAY the elements of stack"<<endl;

cout<<"4. Check if stack is empty"<<endl;

cout<<"5. Check if stack is full"<<endl;

// cout<<"6. Perform linear search in stack"<<endl;

cout<<setw(20)<<"======================="<<endl<<endl;

return;

}

int main()

{

cout<<setw(50)<<endl<<"\*\* STACK IMPLEMENTATION USING ARRAY \*\*"<<endl<<endl;

char ch='y';

int choice,ele, num,res,sz;

stacktype obj;

while(ch=='y' || ch=='Y')

{

operationList();

cout<<"--> Which operation do you want to perform :";

cin>>choice;

cout<<endl;

switch(choice)

{

case 1:

cout<<"\t OPERATION CHOOSEN :: PUSH "<<endl<<endl;

res=obj.isFull();

if (res==1) cout<<"Error : Stack is full, Cannot push elements!"<<endl<<endl;

else

{

cout<<"How many elements do you want to push: ";

cin>>num;

obj.push(num);

}

break;

case 2:

cout<<"\t OPERATION CHOOSEN :: POP"<<endl<<endl;

res=obj.isEmpty();

if (res==1) cout<<"Error : Stack is empty, Cannot pop elements!"<<endl<<endl;

else

{

ele=obj.pop();

cout<<setw(3)<<ele;

}

break;

case 3:

cout<<"\t OPERATION CHOOSEN :: DISPLAY"<<endl<<endl;

obj.display();

break;

case 4:

cout<<"\t OPERATION CHOOSEN :: isEmpty "<<endl<<endl;

res=obj.isEmpty();

if (res==1) cout<<"The stack is Empty"<<endl;

else cout<<"The stack is not Empty"<<endl;

break;

case 5:

cout<<"\t OPERATION CHOOSEN :: isFull"<<endl<<endl;

res=obj.isFull();

if (res==1) cout<<"The stack is full"<<endl;

else cout<<"The stack is not full"<<endl;

break;

default :

cout<<"\t Enter a valid choice!"<<endl;

}

cout<<"\n\n Do you want to continue? (y/n) :";

cin>>ch;

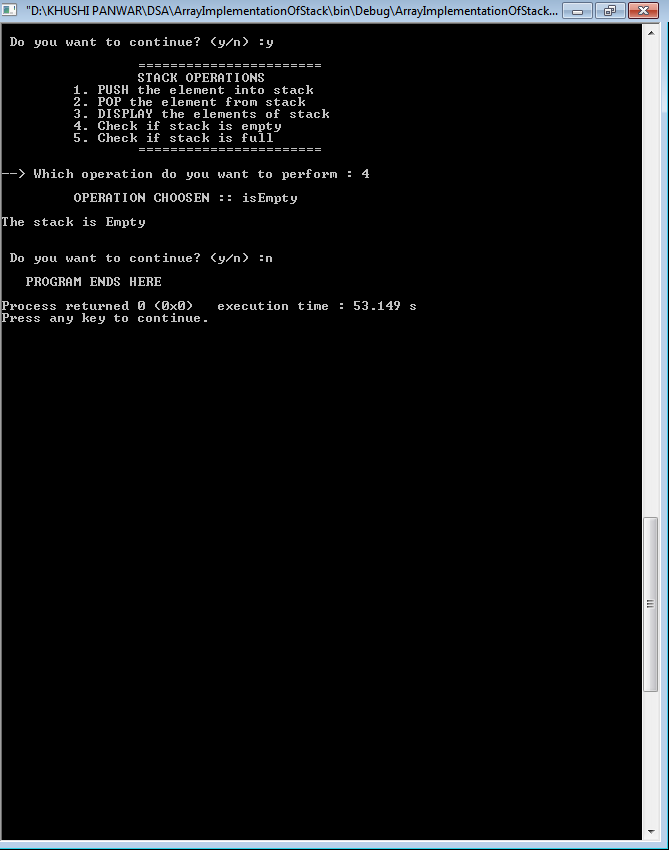
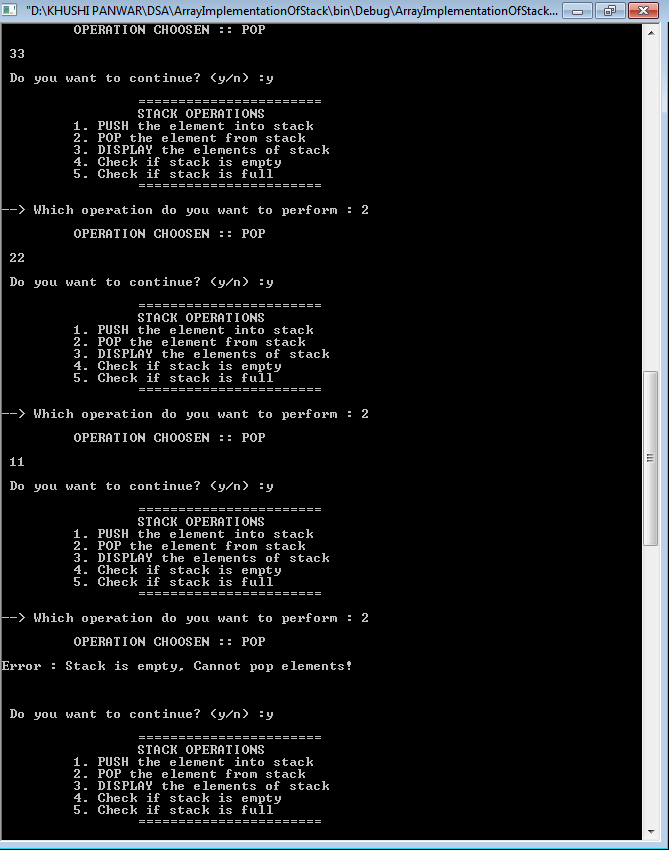
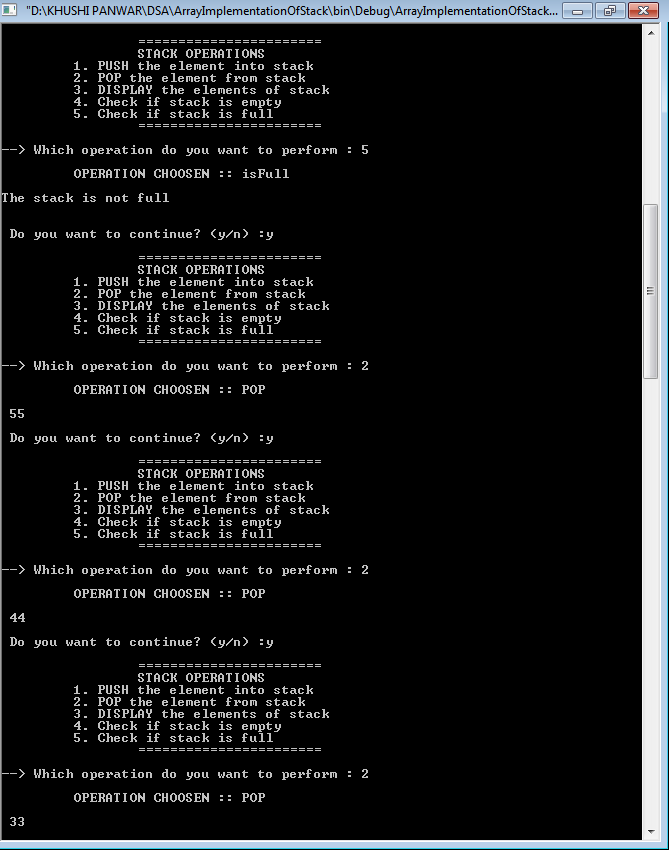
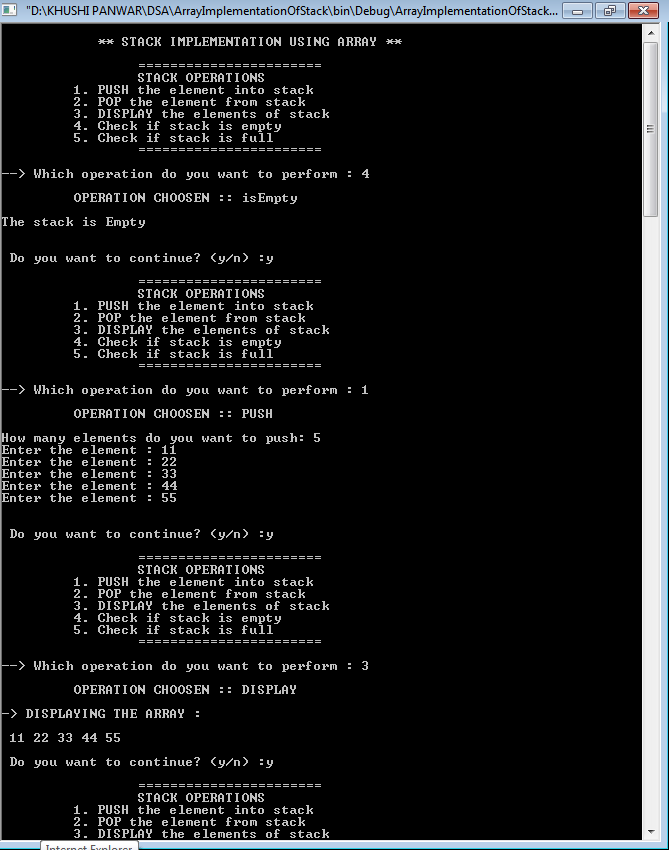
cout<<endl;

}

cout<<setw(20)<<"PROGRAM ENDS HERE"<<endl;

return 0;

}



**6. WAP TO IMPLEMENT STACK USING LINKED LIST**

//implementing stacks using linked lists : "->" is used to refer to the member belonging to the struct "node" type objects.

#include<iostream>

#include <iomanip>

using namespace std;

struct node

{

int info;

node\*next;

};

class stackType

{

node \*top;

public:

stackType()

{

top=NULL;

}

void push(node \*);

int pop();

int isempty();

int display();

node \*createnode(int);

};

int stackType::isempty()

{

if (top==NULL) return 1;

else return -1;

}

node \*stackType::createnode(int x){

node \*temp;

temp=new(node);

temp->info=x;

temp->next=NULL;

return temp;

}

void stackType::push(node \*temp){

if (top==NULL){

top=temp;

}else{

temp->next=top;

top=temp;

}

}

int stackType::display()

{

if(top==NULL) cout<<"-> Stack is empty! "<<endl;

else{

cout<<"-> Contents of stack from top most node are : ";

for (node\* temp=top; temp!=NULL ; temp=temp->next){

cout<<temp->info<<" , ";

}

}

}

int stackType::pop(){

node \*temp;

if (top->next==NULL){

temp=top;

top=NULL;

return temp->info;

delete temp;

}

else{

temp=top;

top=top->next;

return temp->info;

delete temp;

}

}

void menu(){

cout<<setw(30)<<"======================="<<endl;

cout<<"\t\t STACK OPERATIONS"<<endl;

cout<<"\t 1. PUSH the element into stack"<<endl;

cout<<"\t 2. POP the element from stack"<<endl;

cout<<"\t 3. DISPLAY the elements of stack"<<endl;

cout<<"\t 4. Check if stack is empty"<<endl;

cout<<setw(30)<<"\t======================="<<endl<<endl;

return;

}

int main()

{

int choice, ele, val;

char ch='y';

node \*temp1;

stackType st;

do{

menu();

cout<<"-> Select the operation you want to perform : ";

cin>>choice;

switch(choice){

case 1: cout<<endl<<setw(40)<<" \*\* OPERATION CHOOSEN : PUSH the elements \*\* "<<endl;

cout<<"-> Enter a value : ";

cin>>val;

temp1=st.createnode(val);

if (temp1!=NULL){

st.push(temp1);

st.display();

}else{

cout<<"\n Compile out of memory";

}

break;

case 2: cout<<endl<<setw(40)<<" \*\* OPERATION CHOOSEN : POP the elements \*\* "<<endl;

ele=st.pop();

cout<<"-> Element Popped : "<<ele<<endl;

break;

case 3: cout<<endl<<setw(40)<<" \*\* OPERATION CHOOSEN : DISPLAY the elements \*\* "<<endl;

st.display();

break;

case 4: cout<<endl<<setw(40)<<" \*\* OPERATION CHOOSEN : Check emptiness \*\* "<<endl;

ele=st.isempty();

if (ele==1) cout<<endl<<"-> The Stack is Empty! "<<endl;

else cout<<endl<<"-> The Stack is not Empty! "<<endl;

break;

default : cout<<endl<<"!! Select a valid operation !!"<<endl<<endl;

}

cout<<endl<<endl<<"-> Do you want to continue (y/n)?";

cin>>ch;

}while(ch=='y' || ch=='Y');

}

