**Data Structures Practical 3**

**Aim: Implementation of stacks (Using arrays and Linked List)**

1. **Implementation of stack using Array.**

**Code:**

//Stack Implementation with Arrays.

//Date: 18-Mar-2021

#include<iostream>

using namespace std;

void showArray(int arr[],int size) {

//cout<<"Saved Array is \n";

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

cout<<arr[i]<<"\n";

}

}

int main() {

int n,i;

int stackCounter = -1;

int choice;

int new\_value=0;

cout<<"Enter number of elements ";

cin>>n;

int arr[10];

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

cout<<"Enter "<<i+1<<"th Element :";

cin>>arr[i];

stackCounter++;

}

cout<<"Recorded Details \n";

showArray(arr,n);

do{

cout<<"\t1. Push\n\t2. Pop\n\t3. Display Stack\n\t4. Exit\n";

cin>>choice;

switch(choice){

case 1:

if(stackCounter==9){

cout<<"Stack is Full! Cannot add more than 10 elements\n";

}

else{

stackCounter++;

cout<<"Please provide the value to be pushed \n";

cin>>new\_value;

arr[stackCounter]=new\_value;

}

break;

case 2:

if(stackCounter==-1){

cout<<"Stack is Empty!\n";

}

else{

cout<<"Element Popped = "<<arr[stackCounter]<<"\n";

stackCounter--;

}

break;

case 3:

if(stackCounter==-1){

cout<<"Stack is Empty!\n";

}

else{

showArray(arr,stackCounter+1);

}

break;

case 4: break;

default: cout<<"Invalid Data! Please select again.\n";

}

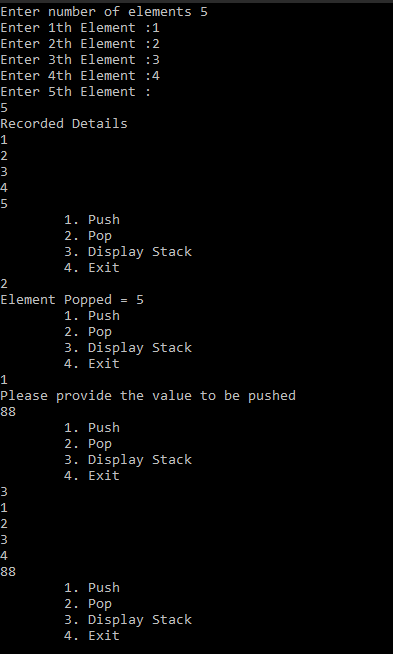
}

while(choice!=4);

return 0;

}

**Output:**



1. **Implementation of stack using LinkedList.**

**Code:**

#include<iostream>

using namespace std;

class Stack\_Linked {

struct node{

int info;

struct node \*link;//A link to the next node

};

struct node \*top;

typedef struct node \*NODE;

public:

Stack\_Linked(){

top=NULL;

}

//function declarations

void push();

void pop();

void display();

};

//This function is to perform the push operation

void Stack\_Linked::push() {

NODE NewNode;

int pushed\_item;

//A new node is created dynamically

NewNode=(NODE)new(struct node);

cout<<"\nInput the new value to be pushed on the stack:";

cin>>pushed\_item;

NewNode->info=pushed\_item;//Data is pushed to the stack

NewNode->link=top;//Link pointer is set to the next node

top=NewNode;//Top pointer is set

}/\*End of push()\*/

//Following function will implement the pop operation

void Stack\_Linked::pop(){

NODE tmp;

if(top == NULL)//checking whether the stack is empty or not

{

cout<<"\nStack is empty\n";

}

else{

tmp=top;//popping the element

cout<<"\nPopped item is:"<<tmp->info;

top=top->link;//resetting the top pointer

tmp->link=NULL;

delete(tmp);//freeing the popped node

}

}/\*End of pop()\*/

//This is to display all the element in the stack

void Stack\_Linked::display() {

if(top==NULL)//Checking whether the stack is empty or not

cout<<"\nStack is empty\n";

else {

NODE ptr=top;

cout<<"\nStack elements:\n";

while(ptr != NULL){

cout<<"\n"<<ptr->info;

ptr = ptr->link;

}/\*End of while \*/

}/\*End of if\*/

}

int main() {

Stack\_Linked So;

int choice;

char opt;

do{

//##MENU##

cout<<"\n\t1.PUSH";

cout<<"\n\t2.POP";

cout<<"\n\t3.DISPLAY";

cout<<"\n\t4.EXIT";

cout<<"\nEnter your choice : ";

cin>>choice;

switch(choice){

case 1:

So.push();//push function is called

break;

case 2:

So.pop();//pop function is called

break;

case 3:

So.display();//display function is called

break;

case 4:

exit(1);

default:

cout<<"\nWrong choice\n";

}

cout<<"\n\nDo you want to continue (Y/y) = ";

cin>>opt;

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

}

**Output:**

