1. You are given an infix expression (mathematical) as input, write a program in C++ which finds the equivalent post-fix expression.

Sample Input 1

a+b-c

Sample Output 1

ab+c-

Sample Input 2

a+b\*c

Sample Output 2

abc\*+

1. Write a function to reverse a string using data structure Stack.
2. Elements can be inserted or deleted only from one end of the stack i.e. from the top. The element at the top is called the top element. The operations of inserting and deleting elements are push() and pop() respectively.

Perform the basic operations of stack using arrays as follows, consider stack as empty.

1 // Selects PUSH operation

2 //Selects POP operation

3 //Display contents of the stack

Here is the stub

#include<bits/stdc++.h>

using namespace std;

#define MAX 6

int Stack[MAX];

int top = -1;

void push(int x);

void pop();

void display();

int main ()

{

int choice, val;

choice = -1;

while(choice !=4)

{

cin >> choice;

switch(choice)

{

case 1:// cout<<”Enter the element to push”;

cin>>val;

push(val);

break;

case 2: // cout<<”Enter the element to pop”;

pop();

break;

case 3:// cout<<”Display the stack elements”;

display();

break;

default:

break;

}

}

return 0;

}

//Add your code here

1. Infix notation is easy to read for humans but needs extra information to make the order of evaluation of the operators clear such as operator precedence, associativity, and brackets ( ). Whereas post-fix notation is easier to parse for a machine and has an advantage over infix that there never arise any questions like operator precedence, brackets(), associativity. Thus generate code that evaluates Postfix expression

**Constraints:** Post-fix expression consisting of operands and operators is input in the form of a string. Operands ranges from 0-9 and arithmetic operators used are +(ASCII value:43),- (ASCII value:45),\*(ASCII value:42),/(ASCII value:47),^(ASCII value:94).

Input: Postfix expression is input in the form of a string Output: Displays the postfix expression evaluated result or display 0 if an expression is wrong. Example 1: 123\*+4- Output: 3

1. N elements are to inserted into a stack which is implemented using an array. You need to find the largest element in the stack.

The stubs push() and stack\_max() needs to be completed.

In the sample Input, the first entry is N the no. of elements to be inserted in the stack, followed by N elements. In the sample output the element which is largest amongst the inserted element is printed.

Sample Input 1:  
5 //no. of elements to be inserted in the stack  
23 45 67 26 78  
Sample Output 1:  
78 //largest element of the stack.

Here is the stub:

#include <map>

#include <set>

#include <list>

#include <cmath>

#include <ctime>

#include <deque>

#include <queue>

#include <stack>

#include <string>

#include <bitset>

#include <cstdio>

#include <limits>

#include <vector>

#include <climits>

#include <cstring>

#include <cstdlib>

#include <fstream>

#include <numeric>

#include <sstream>

#include <iostream>

#include <algorithm>

#include <unordered\_map>

using namespace std;

#define size 20

class stackk

{

int a[10];

int top;

public:

stackk()

{

top=-1;

}

void push(int);

int stack\_max();

};

void stackk::push(int val)

{

//write your code here

}

int stackk::stack\_max()

{

//write your code here

}

int main() {

stackk s;

int n,i,b;

cin>>n;

int m;

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

{

cin>>b;

s.push(b);}

m=s.stack\_max();

cout<<m;

return 0;

}

1. N elements are to inserted into a stack which is implemented using an array. You need to find the smallest element in the stack.

The stubs push() and stack\_min() needs to be completed.

In the sample Input, the first entry is N the no. of elements to be inserted in the stack, followed by N elements. In the sample output the element which is smallest amongst the inserted element is printed.

Sample Input 1:  
5 //no. of elements to be inserted in the stack  
23 45 67 26 78  
Sample Output 1:  
23 //Smallest element of the stack.

Here is the stub

#include <map>

#include <set>

#include <list>

#include <cmath>

#include <ctime>

#include <deque>

#include <queue>

#include <stack>

#include <string>

#include <bitset>

#include <cstdio>

#include <limits>

#include <vector>

#include <climits>

#include <cstring>

#include <cstdlib>

#include <fstream>

#include <numeric>

#include <sstream>

#include <iostream>

#include <algorithm>

#include <unordered\_map>

using namespace std;

#define size 20

class stackk

{

int a[10];

int top;

public:

stackk()

{

top=-1;

}

void push(int);

int stack\_min();

};

void stackk::push(int val)

{

//write your code here

}

int stackk::stack\_min()

{

//write your code here

}

int main() {

stackk s;

int n,i,b;

cin>>n;

int m;

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

{

cin>>b;

s.push(b);}

m=s.stack\_min();

cout<<m;

return 0;

}

1. Pop the element from the stack and display them.

Here is the stub:

#include<iostream>

#define MAX 10

using namespace std;

int STACK[MAX],TOP;

//stack initialization

void initStack(){

TOP=-1;

}

//check it is empty or not

int isEmpty(){

if(TOP==-1)

return 1;

else

return 0;

}

//check stack is full or not

int isFull(){

if(TOP==MAX-1)

return 1;

else

return 0;

}

void push(int num){

if(isFull()){

return;

}

++TOP;

STACK[TOP]=num;

}

//pop - to remove item

void pop()

{

// write your code here

}

int main(){

int num;

int n,i;

initStack();

cin>>n;

i=0;

do{

cin>>num;

push(num);

i++;

}while(i<n);

do

{

pop();

n--;

}while(n>0);

return 0;

}

1. Given a data to be inserted, task is to complete the push() which inserts the data in stack and pop() which delete the data from the stack.

The input's to the functions will be a push() which insert data from the top of the stack and pop() will delete data also from the top of the stack.

With each input data there will be an extra variable that represent which operation will be performed on the stack.

1 - Insert data into the stack

0 - Delete the data from the stack

#include<iostream>

using namespace std;

#define SIZE 10

void push(int);

void pop();

void display();

int stack[SIZE],top=-1;

void push(int value)

{

//write your code

}

void pop()

{

//write your code

}

void display()

{

//write your code

}

int main()

{

int i, j, n, data;

cin>>n;

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

{

cin>>j;

cout<<" " ;

if(j==1)

{

cin>>data;

push(data);

}

if(j==0)

{

pop();

}

}

display();

}

1. Write a program to implement 2 stacks in an array. Both the stacks should use the same array for storing elements.
2. WAP to implement Tower Of Hanoi recursively using Stacks.
3. WAP to implement Tower of Hanoii iteratively using Stack.
4. WAP to implement Stack such that all the operations of PUSH, POP are operated on the middle element of stack. For example, add element in the middle of stack & delete element from the middle of the stack.