**Lets Upgrade - Data Structures And Algorithms**

**Pratyusha Chakravarty**

**Assignment 6**

Question 1

Write a function to find the maximum element in the stack.

Ans.

#include <bits/stdc++.h>

using namespace std;

// A user defined stack that supports getMax() in

// addition to push() and pop()

struct MyStack {

stack<int> s;

int maxElem;

// Prints maximum element of MyStack

void getMax()

{

if (s.empty())

cout << "Stack is empty\n";

// variable maxEle stores the maximum element

// in the stack.

else

cout << "Maximum Element in the stack is: "

<< maxElem << "\n";

}

// Prints top element of MyStack

void peek()

{

if (s.empty()) {

cout << "Stack is empty ";

return;

}

int t = s.top(); // Top element.

cout << "Top Most Element is: ";

// If t < maxElem means maxElem stores

// value of t.

(t > maxElem) ? cout << maxElem : cout << t;

}

// Removing the top element from MyStack

void pop()

{

if (s.empty()) {

cout << "Stack is empty\n";

return;

}

cout << "Top Most Element Removed: ";

int t = s.top();

s.pop();

// Maximum will change as the maximum element

// of the stack is being removed.

if (t > maxElem) {

cout << maxElem << "\n";

maxElem = 2 \* maxElem - t;

}

else

cout << t << "\n";

}

// Removes top element from MyStack

void push(int x)

{

// Inserting new number into the stack

if (s.empty()) {

maxElem = x;

s.push(x);

cout << "Number Inserted: " << x << "\n";

return;

}

// If new number is less than maxElem

if (x > maxElem) {

s.push(2 \* x - maxElem);

maxElem = x;

}

else

s.push(x);

cout << "Number Inserted: " << x << "\n";

}

};

// Main Code

int main()

{

printf("Lets Upgrade - Data Structures And Algorithms \n");

MyStack s;

s.push(3);

s.push(5);

s.getMax();

s.push(7);

s.push(19);

s.getMax();

s.pop();

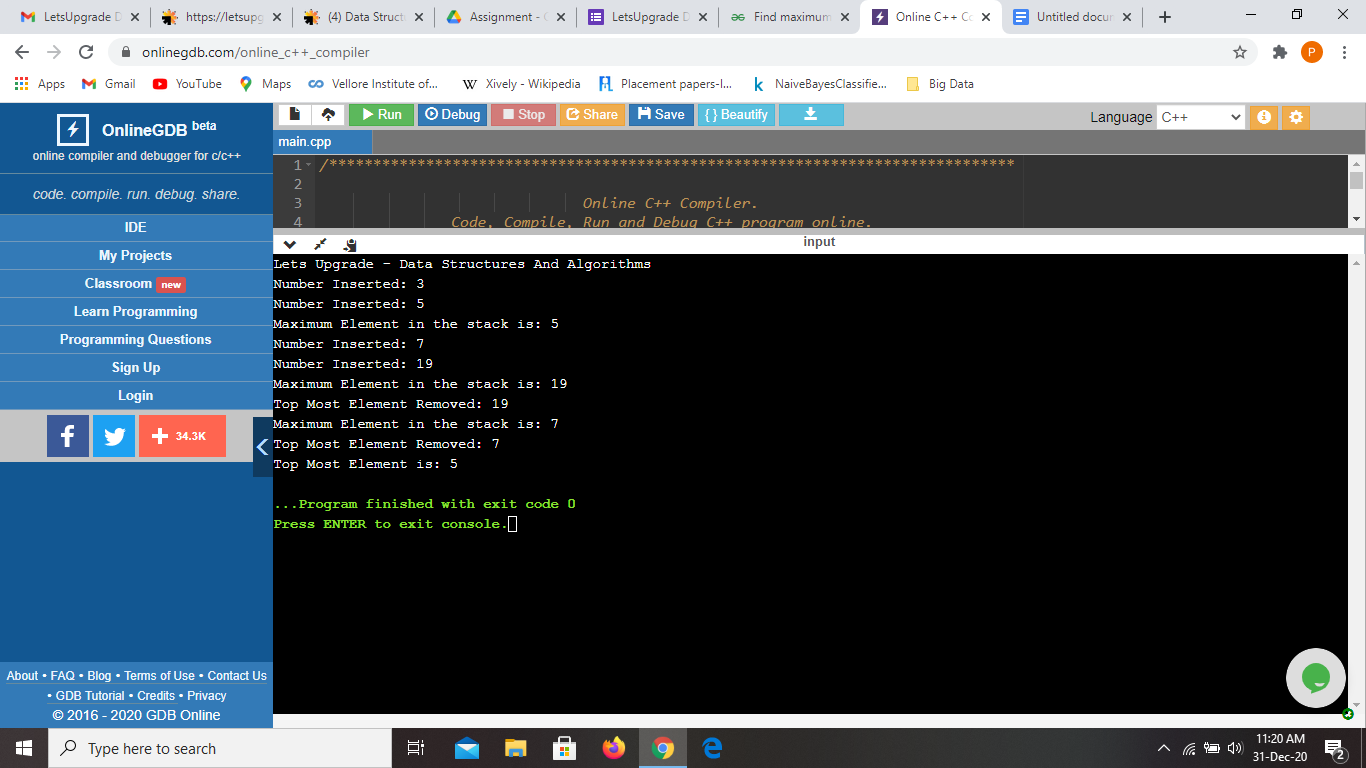
s.getMax();

s.pop();

s.peek();

return 0;

}



Question 2

Write a function to find the minimum element in the stack.

Ans.

#include <bits/stdc++.h>

using namespace std;

// A user defined stack that supports getMin() in

// addition to push() and pop()

struct MyStack

{

stack<int> s;

int minElem;

// Prints minimum element of MyStack

void getMin()

{

if (s.empty())

cout << "Stack is empty\n";

// variable minElem stores the minimum element

// in the stack.

else

cout <<"Minimum Element in the stack is: "

<< minElem << "\n";

}

// Prints top element of MyStack

void peek()

{

if (s.empty())

{

cout << "Stack is empty ";

return;

}

int t = s.top(); // Top element.

cout << "Top Most Element is: ";

// If t < minElem means minElem stores

// value of t.

(t < minElem)? cout << minElem: cout << t;

}

// Removing the top element from MyStack

void pop()

{

if (s.empty())

{

cout << "Stack is empty\n";

return;

}

cout << "Top Most Element Removed: ";

int t = s.top();

s.pop();

// Minimum will change as the minimum element

// of the stack is being removed.

if (t < minElem)

{

cout << minElem << "\n";

minElem = 2\*minElem - t;

}

else

cout << t << "\n";

}

// Removes top element from MyStack

void push(int x)

{

// Inserting new number into the stack

if (s.empty())

{

minElem = x;

s.push(x);

cout << "Number Inserted: " << x << "\n";

return;

}

// If new number is less than minEle

if (x < minElem)

{

s.push(2\*x - minElem);

minElem = x;

}

else

s.push(x);

cout << "Number Inserted: " << x << "\n";

}

};

//Main Code

int main()

{

printf("Lets Upgrade - Data Structures And Algorithms \n");

MyStack s;

s.push(3);

s.push(5);

s.getMin();

s.push(2);

s.push(1);

s.getMin();

s.pop();

s.getMin();

s.pop();

s.peek();

return 0;

}

