DAY 6

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

// C++ program to implement a stack that supports

// getMaximum() in O(1) time and O(1) extra space.

#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 maxEle;

// 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: "

<< maxEle << "\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 < maxEle means maxEle stores

// value of t.

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

}

// Remove 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 > maxEle) {

cout << maxEle << "\n";

maxEle = 2 \* maxEle - t;

}

else

cout << t << "\n";

}

// Removes top element from MyStack

void push(int x)

{

// Insert new number into the stack

if (s.empty()) {

maxEle = x;

s.push(x);

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

return;

}

// If new number is less than maxEle

if (x > maxEle) {

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

maxEle = x;

}

else

s.push(x);

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

}

};

// Driver Code

int main()

{

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;

}

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

// C++ program to implement a stack that supports

// getMinimum() in O(1) time and O(1) extra space.

#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 minEle;

// Prints minimum element of MyStack

void getMin()

{

if (s.empty())

cout << "Stack is empty\n";

// variable minEle stores the minimum element

// in the stack.

else

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

<< minEle << "\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 < minEle means minEle stores

// value of t.

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

}

// Remove 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 < minEle)

{

cout << minEle << "\n";

minEle = 2\*minEle - t;

}

else

cout << t << "\n";

}

// Removes top element from MyStack

void push(int x)

{

// Insert new number into the stack

if (s.empty())

{

minEle = x;

s.push(x);

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

return;

}

// If new number is less than minEle

if (x < minEle)

{

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

minEle = x;

}

else

s.push(x);

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

}

};

// Driver Code

int main()

{

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;

}