DATA STRUCTURES AND ALOGRITHM

Saravanakaruppu. K

BATCH ONE.

Question: 1

Write a function for finding a maximum element in stack

// C++ program to keep track of maximum

// element in a stack

#include <bits/stdc++.h>

Using namespace std;

Class StackWithMax

{

// main stack

Stack<int> mainStack;

// tack to keep track of max element

Stack<int> trackStack;

Public:

Void push(int x)

{

mainStack.push(x);

if (mainStack.size() == 1)

{

trackStack.push(x);

return;

}

// If current element is greater than

// the top element of track stack, push

// the current element to track stack

// otherwise push the element at top of

// track stack again into it.

If (x > trackStack.top())

trackStack.push(x);

else

trackStack.push(trackStack.top());

}

Int getMax()

{

Return trackStack.top();

}

Int pop()

{

mainStack.pop();

trackStack.pop();

}

};

// Driver program to test above functions

Int main()

{

StackWithMax s;

s.push(20);

cout << s.getMax() << endl;

s.push(10);

cout << s.getMax() << endl;

s.push(50);

cout << s.getMax() << endl;

return 0;

}

Question:2

Write a function for finding an minimum element in stack

#include <iostream>

#include <stack>

Class Stack

{

// main stack to store elements

Std::stack<int> st

// variable to store minimum element

Int min;

Public:

// Inserts a given element on top of the stack

Void push(int x)

{

If (s.empty()) {

s.push(x);

min = x;

}

Else if (x > min) {

s.push(x);

}

Else {

s.push(2 \* x – min);

min = x;

}

}

// Removes top element from the stack and returns it

Void pop()

{

If (s.empty()) {

Std::cout << “Stack underflow!!” << ‘\n’;

}

Int top = s.top();

If (top < min)

Min = 2 \* min – top;

s.pop();

}

// Returns the minimum element from the stack in constant time

Int minimum()

{

Return min;

}

};

Int main()

{

Stack s;

s.push(6);

std::cout << s.minimum() << ‘\n’;

s.push(7);

std::cout << s.minimum() << ‘\n’;

s.push(5);

std::cout << s.minimum() << ‘\n’;

s.push(3);

std::cout << s.minimum() << ‘\n’;

s.pop();

std::cout << s.minimum() << ‘\n’;

s.pop();

std::cout << s.minimum() << ‘\n’;

return 0;

}