**DAY 6 ASSIGNMENT**

**Question 1**

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

ANS:

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

#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;

}

Question 2

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

#include "MinStack.h"

#include <iostream>

using namespace std;

void MinStack::push(int d)

{

Node\* t = new Node(d);

if(top == NULL) {

top = min = t;

} else {

t->link = top;

top = t;

// Element inserted is less than minimum

// Set it as the new minimum

if(min->data > top->data) {

top->nextMin = min;

min = top;

}

}

}

int MinStack::pop()

{

// Check for empty stack

if(top == NULL) return -1;

// If dileating the minimum, set the next minimum

if(min == top) {

min = min->nextMin;

}

int d = top->data;

Node\* t = top;

top = top->link;

delete t;

return d;

}

int MinStack::getMinimum()

{

// If Stack is Empty Same check as (top == NULL)

// Since we are not deleting the elements here, we don't need to

// update any pointer, just return the minimum element.

if(min == NULL) {

return -1;

}

else {

return min->data;

}

}

void MinStack::printStack()

{

for (Node\*t = top; t!=NULL; t=t->link) {

cout << " " << t->data;

}

}