**Question 1**

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

#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";

    }

public static void main(String[] args)

{

    MyStack s = new MyStack();

    s.push(3);

    s.push(5);

    s.getMax();

    s.push(7);

    s.push(19);

    s.getMax();

    s.pop();

    s.getMax();

    s.pop();

    s.peek();

    }

}

**Question 2**

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

#include <bits/stdc++.h>

using namespace std;

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

struct MyStack

{

stack<int> s;

int minEle;

void getMin()

{

if (s.empty())

cout << "Stack is empty\n";

else

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

<< minEle << "\n";

}

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;

}

void pop()

{

if (s.empty())

{

cout << "Stack is empty\n";

return;

}

cout << "Top Most Element Removed: ";

int t = s.top();

s.pop();

if (t < minEle)

{

cout << minEle << "\n";

minEle = 2\*minEle - t;

}

else

cout << t << "\n";

}

void push(int x)

{

if (s.empty())

{

minEle = x;

s.push(x);

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

return;

}

if (x < minEle)

{

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

}

else

s.push(x);

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

}

};

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;

}