Day 8

**Question 2**

**Implement push, pop and find the minimum element in a stack in O(1) time complexity.**

#include <iostream>

#include <stdlib.h>

using namespace std;

/\* A simple stack class with basic stack funtionalities \*/ class Stack { private:

static const int max = 100;

int arr[max]; int top;

public:

Stack() { top = -1; } bool isEmpty(); bool isFull(); int pop(); void push(int x);

};

/\* Stack's member method to check if the stack is iempty \*/ bool Stack::isEmpty()

{

if (top == -1)

return true;

return false;

}

/\* Stack's member method to check

if the stack is full \*/ bool Stack::isFull()

{

if (top == max - 1)

return true;

return false;

}

/\* Stack's member method to remove an element from it \*/ int Stack::pop()

{

if (isEmpty()) { cout << "Stack Underflow";

abort();

}

int x = arr[top];

top--; return x;

}

/\* Stack's member method to insert an element to it \*/ void Stack::push(int x)

{

if (isFull()) { cout << "Stack Overflow";

abort();

}

top++;

arr[top] = x;

}

/\* A class that supports all the stack operations and one additional operation getMin() that returns the minimum element from stack at any time. This class inherits from the stack class and uses an auxiliarry stack that holds minimum elements \*/

class SpecialStack : public Stack {

Stack min;

public:

int pop(); void push(int x); int getMin();

};

/\* SpecialStack's member method to insert an element to it. This method makes sure that the min stack is also updated with appropriate minimum values \*/ void SpecialStack::push(int x)

{

if (isEmpty() == true) {

Stack::push(x); min.push(x);

}

else { Stack::push(x); int y = min.pop(); min.push(y); if (x < y) min.push(x);

else

min.push(y);

}

}

/\* SpecialStack's member method to remove an element from it. This method removes top element from min stack also. \*/ int SpecialStack::pop()

{

int x = Stack::pop();

min.pop();

return x;

}

/\* SpecialStack's member method to get minimum element from it. \*/ int SpecialStack::getMin()

{

int x = min.pop(); min.push(x); return x;

}

/\* Driver program to test SpecialStack methods \*/ int main()

{

SpecialStack s;

s.push(10);

s.push(20);

s.push(30);

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

s.push(5);

cout << s.getMin();

return 0;

}