Problem Sheet on Stacks

Date: 26-09-2024

1) Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.

void push(int val) pushes the element val onto the stack.

void pop() removes the element on the top of the stack.

int top() gets the top element of the stack.

int getMin() retrieves the minimum element in the stack.

You must implement a solution with O(1) time complexity for each function.

Example 1:

Input

["push","push","push","getMin","pop","top","getMin"]

[[-2],[0],[-3],[],[],[],[]]

Output

[null,null,null,-3,null,0,-2]

Explanation

minStack.push(-2);

minStack.push(0);

minStack.push(-3);

minStack.getMin(); // return -3

minStack.pop();

minStack.top();   // return 0

minStack.getMin(); // return -2

2) Given a string containing just the characters '(' and ')', return the length of the longest valid (well-formed) parentheses substring

Example 1:

Input: s = "(()"

Output: 2

Explanation: The longest valid parentheses substring is "()".

Example 2:

Input: s = ")()())"

Output: 4

Explanation: The longest valid parentheses substring is "()()".

Example 3:

Input: s = ""

Output: 0

3) Given a stack of integers. The task is to design a special stack such that maximum element can be found in O(1) time and O(1) extra space.

Examples:

Given Stack : 2   5   1   64   --> Maximum

So Output must be 64 when getMax() is called.

4) Write Program which will Evaluate entered Postfix Expression by user using Stack

OUTPUT:

Enter the expression :: 245+*

The result of expression 245+* = 18

5) Given a stack of N numbers and an array of numbers. Count the numbers of pop operations required to get each element of the array. Once an element is popped then it's not pushed back again. Assume that the all the elements from the array present inside the stack initially.

Input: N = 5

Stack: 6 4 3 2 1

Array: 6 3 4 1 2

Output: 1 2 0 2 0