Data Structures and Algorithms Lab
SE-F22
LAB-03

Total Marks: 60 **Issue Date: March 02, 2024**

The objective of this lab is to:
Learning uses of stacks in various problems.

Instructions:
1) Follow the question instructions very carefully, no changes in function prototypes are allowed.
2) Make separate header files for ADTs. (.h_for function prototypes) (.cpp for function implementations)
3) You may only use your own Stack class(implemented as home task), cannot use built in STL Stack.
4) Solve on paper before coding.

In each task try to utilize LIFO behavior of stack, to solve the problem.

Task 01(Balancing Parenthesis) [15Marks]

Given a string `s` consisting only of parentheses `()`, curly braces `{}`, and square brackets `[]`, the task is to determine whether the input string is valid.

An input string is considered valid if it satisfies the following conditions:
- Open brackets must be closed by the same type of brackets.
- Open brackets must be closed in the correct order.
- Every closing bracket must have a corresponding opening bracket of the same type.

Examples:
1) `(()){}` **valid**
2) `{[()]` **not valid**
3) `{{))` **not valid**

Prototype: bool isValid(const String&);

Task 02 (Min Stack) [20 Marks]

Design a class (Min Stack) that supports push, pop, top, and retrieving the minimum element in constant time. Implement the MinStack class:

- MinStack() initializes the stack object.
- void push(int val) pushes the element val onto the stack.
- void pop() removes the element from the top of the stack.
- int top() returns the top element of the stack.
- int getMin() returns the minimum element in the stack.

All calls will be valid. For example top would not be called

You must implement a solution with O (1) time complexity for each function, you may increase the space complexity as you wish.

```
class MinStack {
public:
    MinStack();
    void push(int val);
    void pop();
    int top();
    int getMin();
};
```

Example 1:
push -2
push 0
push -3
getMin (returns -3)
pop
top(returns 0)
getMin (returns -2)

Example 2:
push -5
push 1
push 2
getMin (return -5)
push -6
getMin (return -6)
top (return -6)
pop
top (return 2)
getMin(return -5)

Task 03 (Next Greater Element) [25 Marks]

Given an integer array you have to find the nearest greater element or next greater to the left of each element. If there is no next greater element to the left then store -1 for that.

Input: [4, 5, 2, 0]
Output: [-1, -1, 5, 2]

Input: [1, 6, 4, 10, 2, 5]
Output: [-1, -1, 6, -1, 10, 10]

No Cheating ☺