Assignment 2: Arrays, Stacks and Queues

Objective:

The main goal of the assignment is to get familiar with the stack and queue data structures and get practice while implementing. Along with it, you will get to explore these data structures in the context of their applications.

Part A *(25 marks)*

Problem Statement:

Implement the *stack* data structure. You must implement the class **edu.iitd.col1062020.MyStack<T>** with the following methods. It must be a generic class (i.e. you can specify the data type while creating the object). You are being provided the Interface with signatures of the functions and you need to complete the code to implement the desired functionality.

1. **MyStack():** Constructor for **MyStack** class. Initialize an empty array of type T.
2. **void push(T value)**: Insert the value at the top of the stack. Resize if the internal array is full.
3. **T pop() throws edu.iitd.col1062020.EmptyStackException:** Delete the top element of the stack, and return it. If the stack is empty, you must throw the specified exception.
4. **T top() throws edu.iitd.col1062020.EmptyStackException:** Return the top element of the stack. If the stack is empty, you must throw the specified exception.
5. **boolean isEmpty():** Returns true if the stack is empty, false otherwise.

You are **NOT** allowed to use any inbuilt implementation of stack.

Input Constraints:

MyStack should be capable of handling upto 10^6 values. You could initialize the internal array to a fixed size in the constructor or could resize and expand the array when it is full (latter is recommended).

Part B *(35 marks)*

Problem Statement:

In this part you will use your above implementation to implement a simple calculator. You will be given a **String** representing a valid arithmetic expression, and you will have to evaluate the expression and find the result. The expression **String** may contain open (and closing parentheses), the plus +, minus - or multiplication * sign, non-negative integers and empty spaces. You may assume that the given expression is always valid.

You must implement the class **edu.iitd.col1062020.MyCalculator** with the following methods. As before you are provided the Interface with signatures of the functions and you need to complete the code.

1. **MyCalculator():** Constructor for the **MyCalculator** class.
2. **int calculate(String expression):** Find the value of the expression using the **MyCalculator** class implemented above.

Example 1:

```
Input: "1 + 10"
Output: 11
```

Example 2:

```
Input: " 2-1 + 2 "
Output: 3
```

Example 3:

```
Input: "(1+(4+5+2)-3)+(6+18)"
Output: 33
```

Example 4:

```
Input: "4 * 2 + 2 * (2 + 9)"
Output: 30
```

You are **NOT** allowed to use any inbuilt function for expression evaluation.

Input Constraints:

1 <= Length of Expression (excluding spaces) <= 2 * 10^5

Part C *(40 marks)*

Problem Statement:

In this part, you will use the stack implementation from Part A to sort a given array. You are given an array of integers and you have to sort it using a single stack *A*. You are allowed to only push elements from the given array of integers (treating it as a queue), and your sequence of pops should form a sorted sequence of the given array. You have to return the sequence of operations (described below) in order to sort the elements.

Allowed operations:

1. Extract *one* input element (with least index) and insert in the stack. Represented as PUSH.
2. Pop the top-most element of the stack. Represented as POP.
3. If it is not possible to sort the given numbers with a single stack, then NOTPOSSIBLE.

Return the sequence of operations as an Array of **String**. Your sequence should correspond to a valid sequence of operations, and the pop sequence should form a sorted array of given numbers (in ascending order). Your number of PUSH and POP should be exactly equal to the length of the given array of numbers.

You must implement the class **edu.iitd.col1062020.StackSort** with the following methods. As before you are provided the Interface with signatures of the functions and you need to complete the code.

1. **StackSort():** Constructor for the **StackSort** class.
2. **String[] sort(int[] nums):** Sort the **nums** array, using single stack, and return the sequence of operations.

Example 1:

```
Input: [10, 702, 36, 125, 82]
Output: ["PUSH", "POP", "PUSH", "PUSH", "POP", "PUSH", "PUSH", "POP", "POP",
"POP"]
Explanation: The POP sequence forms the list [10, 36, 82, 125, 702] which is
sorted.
```

Example 2:

```
Input: [1023, 5029, 158]
Output: ["NOTPOSSIBLE"]
```

Input Constraints:

1 <= length of input array <= 1024

-10^9 <= input values <= 10^9

Input array could possibly contain duplicate values

Part D (Bonus) *(10 marks)*

Problem Statement:

In this part, you will extend the above implementation, to sort the array by repeatedly using the stack. Let us denote the operation applied in Part C by f, then on applying f to the input array of numbers, you get another array of numbers, on which you can reapply this f. You are allowed to do the above procedure as many times as you like, provided it results in a sorted array at the end. You need to find the least k, such that applying the operation f, k times leads to the sorted array.

Return the sequence of operations as an Array of Array of **String**, with the outer Array of length k, the least number of operations in which you can sort the input array of given numbers. Your sequence should correspond to a valid sequence of operations, and the sequence should form a sorted array of given numbers (in ascending order) at the end of the last operation.

Add the following function to the class **edu.iitd.col1062020.StackSort** for this purpose:

1. **String[][] kSort(int[] nums):** Sort the **nums** array, using k operations, and return the sequence of operations.

Example 1:

```
Input: [1023, 5029, 158]
Output: [[“PUSH”, “POP”, “PUSH”, “PUSH”, “POP”, “POP”],[“PUSH”,  “PUSH”,
“PUSH”, “POP”, “POP”, “POP”], [“PUSH”, “PUSH”, “POP”, “PUSH”, “POP”, “POP”]]

Explanation: The sequence of numbers at the end of each sequence:
   1. [1023, 158, 5029]
   2. [5029, 158, 1023]
   3. [158, 1023, 5029]
```

Note: Above is a valid sequence of operations to sort the initial list of numbers. However, it could be done with k < 3. Hence the given output will **NOT** be treated as a PASS, and you will have to do it with k < 3.

What to submit?

1. Submit your code in a *.zip* file named in the format **<EntryNo>.zip**. Make sure that when we run unzip <yourfile>.zip a folder <YourEntryNo> should be produced in the current working directory. The interface files are provided in a *zip* file exactly in the submission format.
   You will be penalized for any submissions that do not conform to this requirement.
2. The exact directory structure will be detailed on the moodle submission link. Your submission will be auto-graded. This means that it is essential to make sure that your code follows the specifications of the assignment precisely.
3. Your code must compile and run on our VMs. They run amd64 Linux version ubuntu 16.04 running Java JDK 11.0.5 (https://www.oracle.com/technetwork/java/javase/downloads/jdk11-downloads-5066655.html).
4. You will be able to check if your submission complies with the proper format and runs as expected in the test environment by submitting to Moodle. The auto-grading in Moodle will run your code and verify if it passes the preliminary test cases.
5. Your submission will only be accepted once it passes the initial test cases. For this, you must at least complete the Minimal submission tests of the assignment (will be announced soon). Further test cases will be used for Evaluation.

What is Allowed / Not Allowed?

1. You must work on this assignment individually.
2. The programming language to be used is Java.
3. You are not allowed to use any other external libraries for this assignment.
4. Your code must be your own. You are not to take guidance from any general-purpose or problem-specific code meant to solve this or related problem.
5. We will run a plagiarism detection check. Any person found guilty will be awarded a suitable penalty as announced in the class/course website. Please read academic integrity guidelines on the course home page and follow them carefully.