[CO1]

[Each method carries 5 marks]

Instructions for students:

- **SUBMISSION IN THE SAME WEEK**
- All your methods must be written in one single .java or .py or .pynb file. DO NOT CREATE separate files for each task.
- If you are using PYTHON, then follow the coding templates shared in this folder.

NOTE:

- **YOUR CODE SHOULD WORK FOR ANY VALID INPUTS. [Make changes to the Sample Inputs and check whether your program works correctly]**
- **ALL YOUR TASKS SHOULD BE OUTSIDE CLASS**

Tasks on Stack:

1. Parenthesis Balancing:

Your younger sister is learning arithmetics and came across several types of parentheses for the first time. She finds them pretty exciting and proceeds to put as much parenthesis as she wants in a mathematical statement. She came to show you her statements and after watching lots of parentheses here and there, you decided to write a program that checks whether the parentheses are maintained properly, that is, the pairs and the orders of parentheses are balanced in expression or not.

You need to solve the above problem using **Stack class**. You cannot use other methods than pop(), peek(), push(), isEmpty() methods of Stack.

Sample Input String	Sample Output
1+2*[3*3+{4–5(6(7/8/9)+10)–11+(12*8)]+14	Unbalanced
([A+B]-C)/{D*E}+[2*[(2A+5){5B}]-{7C-9AB}]	Balanced
(A+B)-C)	Unbalanced
[10*[3-(5-2)]	Unbalanced

2. Black Diamond:

Faisal is working in a diamond mine, trying to extract the highest number of diamonds "< >". A "<" followed by ">" forms one new diamond. He must exclude all the sand particles found (denoted by ".") and **unpaired** "<", ">" in this process to extract new diamonds.

You need to solve the above problem using **Stack class**. You cannot use other methods than pop(), peek(), push(), isEmpty() methods of Stack.

Complete the function **count_diamond** which will take an object of Stack and a string and then return the number of diamonds that can be extracted using the mentioned process.

Sample Input String	Sample Output
.<...<<..>>....>....>>>.	3
<<<<..<.....<<<<....>	1
<..><.<..>>	3

Explanation:
For an input ".<...<<..>>....>....>>>." three diamonds are formed. The first is taken from <..> resulting ".<...<>....>....>>>." The second diamond <> is then removed, leaving ".<.....>....>>>." The third diamond <> is then removed, leaving at the end ".....>>>." without the possibility of extracting new diamonds. Hence, 3 diamonds have been extracted.

BONUS QUESTION:

3. Tower of God:

A kid is trying to make a tower using blocks with numbers or characters written on them. He puts a block on the top of another to make the tower. As for removing the blocks, he removes the topmost block if needed. He is afraid that removing other blocks from the middle in one go will result in the collapse of his tower.

Now his friend wants your nth block from the top of your tower. He is willing to give him the nth block from the top, preserving the relative position of others.

Following his tower build process ,solve this scenario using **Stack** class. You can only use the stack methods.

In the examples, consider the rightmost element to be the topmost element.

Sample Tower	Sample Output	Explanation
Stack: 4 19 23 17 5 n: 2	Stack: 4 19 23 5	In this tower, the 2nd block from the top is the block with 17. After removing the block the stack looks like 4 19 23 5
Stack: 73 85 15 41 n: 3	Stack: 73 15 41	In this tower, the 3rd block from the top is the block with 85. After removing the block the stack looks like 73 15 41