# Question 1

Given a string **A** of parantheses ‘(‘ or ‘)’.

The task is to find minimum number of parentheses ‘(‘ or ‘)’ (at any positions) we must add to make the resulting parentheses string valid.

An string is valid if:

* Open brackets must be closed by the corresponding closing bracket.
* Open brackets must be closed in the correct order.

**Input Format**

First and only argument is an string **A**.

**Output Format**

Return a single integer denoting the **minimum** number of parentheses ‘(‘ or ‘)’ (at any positions) we must add in **A** to make the resulting parentheses string valid.

**Example Input**

Input 1:

A = "())"

Input 2:

A = "((("

**Example Output**

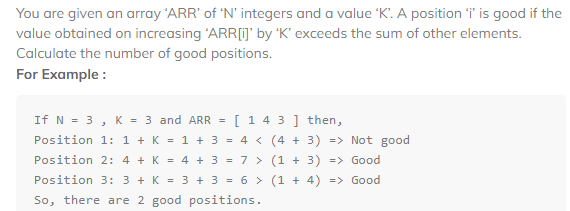
Output 1:

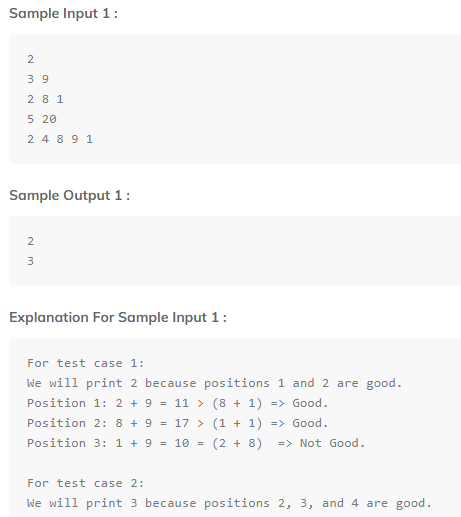
1

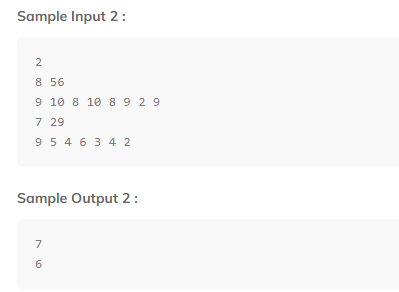
Output 2:

3

Question 2







Question 3

In a long hallway some soldiers are walking from left to right and some from right to left all at the same speed.  
Every time while walking they cross through another soldier they salute and move ahead.  
Given a string **A** of length **N** showing the soldiers' direction they are walking. **'<'** denotes a soldier is walking from right to left, and **'>'** denotes a soldier is walking from left to right. Return the number of Salutes done.

**Problem Constraints**

1 <= N <= 105  
A = {'<', '>'}

**Input Format**

The first argument is a string A.

**Output Format**

Return a single integer denoting the number of salutes done.

**Example Input**

Input 1:

A = '>>><<<'

Input 2:

A = '<>'

**Example Output**

Output 1:

9

Output 2:

0

**Example Explanation**

Explanation 1:

Soldier 1 will salute with 4, 5, 6. Same goes for soldier 2 and 3.

Hence, the total number of salutes is 9.

Explanation 2:

There will be no salutes as no two soldiers will cross each other.

Question 4

Given an array of integers **A**, sort the array into a wave-like array and return it.   
In other words, arrange the elements into a sequence such that

a1 >= a2 <= a3 >= a4 <= a5.....

**NOTE:** If multiple answers are possible, return the lexicographically smallest one.

**Problem Constraints**

1 <= len(A) <= 106  
1 <= A[i] <= 106

**Input Format**

The first argument is an integer array A.

**Output Format**

Return an array arranged in the sequence as described.

**Example Input**

Input 1:

A = [1, 2, 3, 4]

Input 2:

A = [1, 2]

**Example Output**

Output 1:

[2, 1, 4, 3]

Output 2:

[2, 1]

**Example Explanation**

Explanation 1:

One possible answer : [2, 1, 4, 3]

Another possible answer : [4, 1, 3, 2]

First answer is lexicographically smallest. So, return [2, 1, 4, 3].

Explanation 1:

Only possible answer is [2, 1].

Question 5

You are given an integer array **A**.  
You have to find the number of occurences of each number.  
Return an array containing only the occurences with the smallest value's occurence first.  
For example, A = [4, 3, 3], you have to return an array [2, 1], where 2 is the number of occurences for element 3, and 1 is the number of occurences for element 4. But, 2 comes first because 3 is smaller than 4.

**Problem Constraints**

1 <= |A| <= 105  
1 <= Ai <= 109

**Input Format**

The first argument is the integer array A.

**Output Format**

Return an integer array denoting the occurences of each number.

**Example Input**

Input 1:

A = [1, 2, 3]

Input 2:

A = [4, 3, 3]

**Example Output**

Output 1:

[1, 1, 1]

Output 2:

[2, 1]

**Example Explanation**

Explanation 1:

All the elements occur once, so the resultant array should be [1, 1, 1].

Explanation 2:

Explained in the description above.

Question 6

Given an integer array **A**, move all 0's to the end of it while maintaining the relative order of the non-zero elements.

Note that you must do this in-place without making a copy of the array.

**Problem Constraints**

1 <= |A| <= 105

**Input Format**

First argument is array of integers A.

**Output Format**

Return an array of integers which satisfies above property.

**Example Input**

Input 1:

A = [0, 1, 0, 3, 12]

Input 2:

A = [0]

**Example Output**

Ouput 1:

[1, 3, 12, 0, 0]

Ouput 2:

[0]

**Example Explanation**

Explanation 1:

Shift all zeroes to the end.

Explanation 2:

There is only one zero so no need of shifting.