1. Problem Statement – An automobile company manufactures both a two wheeler (TW) and a four wheeler (FW). A company manager wants to make the production of both types of vehicle according to the given data below:

- 1st data, Total number of vehicle (two-wheeler + four-wheeler)=v
- 2nd data, Total number of wheels = W

The task is to find how many two-wheelers as well as four-wheelers need to manufacture as per the given data.

Example:
Input:
200 -> Value of V
540 -> Value of W

Output:
TW =130 FW=70

Explanation:
130+70 = 200 vehicles
(70*4)+(130*2)= 540 wheels

Constraints:

- 2<=W
- W%2=0
- V<W
- Print "INVALID INPUT" , if inputs did not meet the constraints.

Written program code should generate two outputs, each separated by a single space character(see the example)
Additional messages in the output will result in the failure of test case.
==

2. Problem Statement – Given a string S(input consisting) of '*' and '#'. The length of the string is variable. The task is to find the minimum number of '*' or '#' to make it a valid string. The string is considered valid if the number of '*' and '#' are equal. The '*' and '#' can be at any position in the string.

Note: The output will be a positive or negative integer based on number of '*' and '#' in the input string.

(*>#): positive integer
(#>*): negative integer
(#=*): 0

Example 1:
Input:
###*** -> Value of S

Output:
0 → number of * and # are equal
==

3. A party has been organised on cruise. The party is organised for a limited time(T). The number of guests entering (E[i]) and leaving (L[i]) the party at every hour is represented as elements of the array. The task is to find the maximum number of guests present on the cruise at any given instance within T hours.

Example 1:
Input:

5 -> Value of T
[7,0,5,1,3] -> E[], Element of E[0] to E[N-1], where input each element is separated by new line
[1,2,1,3,4] -> L[], Element of L[0] to L[N-1], while input each element is separate by new line.

Output:
8 -> Maximum number of guests on cruise at an instance.

Explanation:
1st hour:
Entry: 7 Exit: 1
No. of guests on ship: 6

2nd hour:
Entry: 0 Exit: 2
No. of guests on ship: 6-2=4

Hour 3:
Entry: 5 Exit: 1
No. of guests on ship: 4+5-1=8

Hour 4:
Entry: 1 Exit: 3
No. of guests on ship: 8+1-3=6

Hour 5:

Entry: 3 Exit: 4
No. of guests on ship: 6+3-4=5
Hence, the maximum number of guests within 5 hours is 8.

Example 2:
Input:
4 -> Value of T
[3,5,2,0]   -> E[], Element of E[0] to E[N-1], where input each element is separated by new line.
[0,2,4,4]   -> L[], Element of L[0] to L[N-1], while input each element in separated by new line

Output:
6
Cruise at an instance

Explanation:
Hour 1:
Entry: 3 Exit: 0
No. of guests on ship: 3

Hour 2:
Entry: 5 Exit: 2
No. of guest on ship: 3+5-2=6

Hour 3:
Entry: 2 Exit: 4
No. of guests on ship: 6+2-4= 4

Hour 4:
Entry: 0 Exit: 4
No. of guests on ship: 4+0-4=0

Hence, the maximum number of guests within 5 hours is 6.
==
4. You are given n balloons, indexed from 0 to n - 1. Each balloon is painted with a number on it represented by an array nums. You are asked to burst all the balloons.
If you burst the ith balloon, you will get nums[i - 1] * nums[i] * nums[i + 1] coins. If i - 1 or i + 1 goes out of bounds of the array, then treat it as if there is a balloon with a 1 painted on it.
Return the maximum coins you can collect by bursting the balloons wisely.

Example 1:
Input: nums = [3,1,5,8]
Output: 167
Explanation:
nums = [3,1,5,8] --> [3,5,8] --> [3,8] --> [8] --> []
coins = 3*1*5 + 3*5*8 + 1*3*8 + 1*8*1 = 167

Example 2:
Input: nums = [1,5]
Output: 10
==
5. You have n boxes. You are given a binary string boxes of length n, where boxes[i] is '0' if the ith box is empty, and '1' if it contains one ball.
In one operation, you can move one ball from a box to an adjacent box. Box i is adjacent to box j if abs(i - j) == 1. Note that after doing so, there may be more than one ball in some boxes.
Return an array answer of size n, where answer[i] is the minimum number of operations needed to move all the balls to the ith box.
Each answer[i] is calculated considering the initial state of the boxes.

Example 1:
Input: boxes = "110"
Output: [1,1,3]
Explanation: The answer for each box is as follows:
1) First box: you will have to move one ball from the second box to the first box in one operation.
2) Second box: you will have to move one ball from the first box to the second box in one operation.
3) Third box: you will have to move one ball from the first box to the third box in two operations, and move one ball from the second box to the third box in one operation.

Example 2:
Input: boxes = "001011"
Output: [11,8,5,4,3,4]
==