**LAB2 Problems**

**1.There are n gas stations along a circular route, where the amount of gas at the ith station is gas[i].You have a car with an unlimited gas tank and it costs cost[i] of gas to travel from the ith station to its next (i + 1)th station. You begin the journey with an empty tank at one of the gas stations.  
Given two integer arrays gas and cost, return the starting gas station’s index if you can travel around the circuit once in the clockwise direction, otherwise return -1. If there exists a solution, it is guaranteed to be unique**

Input: gas = [1,2,3,4,5], cost = [3,4,5,1,2]  
Output: 3  
Explanation:  
Start at station 3 (index 3) and fill up with 4 unit of gas. Your tank = 0 + 4 = 4  
Travel to station 4. Your tank = 4 - 1 + 5 = 8  
Travel to station 0. Your tank = 8 - 2 + 1 = 7  
Travel to station 1. Your tank = 7 - 3 + 2 = 6  
Travel to station 2. Your tank = 6 - 4 + 3 = 5  
Travel to station 3. The cost is 5. Your gas is just enough to travel back to station 3.  
Therefore, return 3 as the starting index.

Input: gas = [2,3,4], cost = [3,4,3]  
Output: -1  
Explanation:  
You can’t start at station 0 or 1, as there is not enough gas to travel to the next station.  
Let’s start at station 2 and fill up with 4 unit of gas. Your tank = 0 + 4 = 4  
Travel to station 0. Your tank = 4 - 3 + 2 = 3  
Travel to station 1. Your tank = 3 - 3 + 3 = 3  
You cannot travel back to station 2, as it requires 4 unit of gas but you only have 3.  
Therefore, you can’t travel around the circuit once no matter where you start.

**2.Given an expression string exp, write a program to examine whether the pairs and the orders of “{“, “}”, “(“, “)”, “[“, “]” are correct in the given expression.**

**Example:**

***Input: exp = “[()]{}{[()()]()}”   
Output: Balanced  
Explanation: all the brackets are well-formed***

***Input: exp = “[(])”   
Output: Not Balanced   
Explanation: 1 and 4 brackets are not balanced because   
there is a closing ‘]’ before the closing ‘(‘***

# **3.Longest Common Subsequence (LCS)**

**Given two strings text1 and text2, return the length of their longest common subsequence. If there is no common subsequence, return 0.**

**A subsequence of a string is a new string generated from the original string with some characters (can be none) deleted without changing the relative order of the remaining characters.**

**For example, "ace" is a subsequence of "abcde".**

**A common subsequence of two strings is a subsequence that is common to both strings.**

Example 1:

Input: text1 = "abcde", text2 = "ace"

Output: 3

Explanation: The longest common subsequence is "ace" and its length is 3.

Example 2:

Input: text1 = "abc", text2 = "abc"

Output: 3

Explanation: The longest common subsequence is "abc" and its length is 3.

Example 3:

Input: text1 = "abc", text2 = "def"

Output: 0

Explanation: There is no such common subsequence, so the result is 0.