1. Count nice pairs in array
   You are given an array nums that consists of non-negative integers. Let us define rev(x) as the reverse of the non-negative integer x. For example, rev(123) = 321, and rev(120) = 21. A pair of indices (i, j) is nice if it satisfies all of the following conditions:

   0 <= i < j < nums.length
   nums[i] + rev(nums[j]) == nums[j] + rev(nums[i])
   Return the number of nice pairs of indices. Since that number can be too large, return it modulo 109 + 7.
   Example 1: Input: nums = [42,11,1,97] Output: 2
   Explanation: The two pairs are:
   - (0,3) : 42 + rev(97) = 42 + 79 = 121, 97 + rev(42) = 97 + 24 = 121.
   - (1,2) : 11 + rev(1) = 11 + 1 = 12, 1 + rev(11) = 1 + 11 = 12.
   Example 2: Input: nums = [13,10,35,24,76] Output: 4

2. Ransom Note.
   Given two strings ransomNote and magazine, return true if ransomNote can be constructed by using the letters from magazine and false otherwise.

   Each letter in magazine can only be used once in ransomNote.
   Example 1: Input: ransomNote = "a", magazine = "b” Output: false

   Example 2: Input: ransomNote = "aa", magazine = "ab" Output: false
   Example 3:

   Input: ransomNote = "aa", magazine = "aab" Output: true

3. Gas Station.
   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
   Example 1: 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.
   Example 2: 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.

4. Text Justification

Given an array of strings words and a width maxWidth, format the text such that each line has exactly maxWidth characters and is fully (left and right) justified.

You should pack your words in a greedy approach; that is, pack as many words as you can in each line. Pad extra spaces ' ' when necessary so that each line has exactly maxWidth characters.

Extra spaces between words should be distributed as evenly as possible. If the number of spaces on a line does not divide evenly between words, the empty slots on the left will be assigned more spaces than the slots on the right. For the last line of text, it should be left-justified, and no extra space is inserted between words.

Note: A word is defined as a character sequence consisting of non-space characters only.

Each word's length is guaranteed to be greater than 0 and not exceed maxWidth.

The input array words contains at least one word.

Example 1: Input: words = ["This", "is", "an", "example", "of", "text", "justification."], maxWidth = 16

Output:

[

"This is an",

"example of text",

"justification. "

]Example 2: Input: words = ["What","must","be","acknowledgment","shall","be"], maxWidth = 16

Output:

[

"What must be",

"acknowledgment ",

"shall be "

]

Explanation: Note that the last line is "shall be " instead of "shall be", because the last line must be left-justified instead of fully-justified.

Note that the second line is also left-justified because it contains only one word.

Example 3: Input: words = ["Science","is","what","we","understand","well","enough","to","explain","to","a","computer.","Art","is","everything","else","we","do"], maxWidth = 20

Output:

[

"Science is what we",

"understand well",

"enough to explain to",

"a computer. Art is",

"everything else we",

"do "

]

5.