Practice Test

Instructions to be followed:

- Should not use any built-in functionalities or Collection Interface.
- You can solve the questions in any order.
- Read the question properly and do the problem.
- Note down the start time and end time of each question with evaluator signature.
- If any of the test cases gets failed, that question never be counted as completed one.
- Turn off your net connection, after downloading the question paper.
- Submit the timing sheet before leaving.

----

1. Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M.

Symbol	Value
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

For example, 2 is written as II in Roman numeral, just two ones added together. 12 is written as XII, which is simply X + II. The number 27 is written as XXVII, which is XX + V + II.
Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not IIII. Instead, the number four is written as IV. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as IX.

There are six instances where subtraction is used:

- ✗ I can be placed before V (5) and X (10) to make 4 and 9.
- ✗ X can be placed before L (50) and C (100) to make 40 and 90.
- ✗ C can be placed before D (500) and M (1000) to make 400 and 900. Given a roman numeral, convert it to an integer.

Example 1:
Input: s = "III"
Output: 3
Explanation: III = 3.

Example 2:
Input: s = "LVIII"
Output: 58
Explanation: L = 50, V= 5, III = 3.

Example 3:
Input: s = "MCMXCIV"
Output: 1994
Explanation: M = 1000, CM = 900, XC = 90 and IV = 4.

===

2. Given n pairs of parentheses, write a function to generate all combinations of well-formed parentheses.
Example 1: Input: n = 3
Output: ["((()))","(()())","(())()","()(())","()()()"]

Example 2: Input: n = 1
Output: ["()"]

Constraints:
1 <= n <= 8

===

3. Given a string s and an array of strings contains words of the same length. Return all starting indices of substring(s) in s that is a concatenation of each word in words exactly once, in any order, and without any intervening characters. You can return the answer in any order.
Example 1:
Input: s = "barfoothefoobarman", words = ["foo","bar"]
Output: [0,9]
Explanation: Substrings starting at index 0 and 9 are "barfoo" and "foobar" respectively. The output order does not matter, returning [9,0] is fine too.

Example 2:
Input: s = "wordgoodgoodgoodbestword", words = ["word","good","best","word"]
Output: []

Example 3:
Input: s = "barfoofoobarthefoobarman", words = ["bar","foo","the"]
Output: [6,9,12]

Constraints:
1 <= s.length <= 104
s consists of lower-case English letters.
1 <= words.length <= 5000
1 <= words[i].length <= 30
words[i] consists of lower-case English letters.
==
4. Given a sorted array of distinct integers and a target value, return the index if the target is found. If not, return the index where it would be if it were inserted in order.
Example 1:
Input: nums = [1,3,5,6], target = 5
Output: 2

Example 2:
Input: nums = [1,3,5,6], target = 2
Output: 1

Example 3:
Input: nums = [1,3,5,6], target = 7
Output: 4

Constraints:
1 <= nums.length <= 100
0 <= nums[i] <= 100
nums contains distinct values sorted in ascending order.
0 <= target <= 100
==
5. Given a positive integer n, return the nth term of the count-and-say sequence.
Example 1: For example, the saying and conversion for digit string "3322251":- 3322251 - given two 3's, three 2's, one 5 and one 1
23+32+15+11
23321511

Example 2:
1211
one 1, one 2,two 1's
11+12+21
111221
==