PROBLEM SOLVING QUESTIONS

Q.1 Write a program to find the median of the array obtained after merging the A and B arrays. Suppose that 2 sorted arrays A and B of size n each. Input must be taken from the user.

Q.2 Write a program to find the number occurring odd number of times from the array. For array values can be taken from the user.

Q.3 Write a program to find a peak element from an array which is not smaller than its neighbours.

Q.4 Write a program to check if the given pair of string are anagram or not.
[HINT: An Anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.]

Q.5 Write a code to print the following pattern:
4*4*4*4
3*3*3
2*2
1
1
2*2
3*3*3
4*4*4*4

Q.6 Given an integer array nums, reorder it such that nums[0] < nums[1] > nums[2] < nums[3]....
You may assume the input array always has a valid answer.
Example:
Input: nums = [1,3,2,2,3,1]
Output: [2,3,1,3,1,2]

Constraints:
1 <= nums.length <= 5 * 104
0 <= nums[i] <= 5000

Q.7 There are n cars going to the same destination along a one-lane road. The destination is target miles away.

You are given two integer array position and speed, both of length n, where position[i] is the position of the ith car and speed[i] is the speed of the ith car (in miles per hour).

A car can never pass another car ahead of it, but it can catch up to it and drive bumper to bumper at the same speed. The faster car will slow down to match the slower car's speed. The distance between these two cars is ignored (i.e., they are assumed to have the same position).

[HINT: A car fleet is some non-empty set of cars driving at the same position and same speed. Note that a single car is also a car fleet. If a car catches up to a car fleet right at the destination point, it will still be considered as one car fleet.]

Return the number of car fleets that will arrive at the destination.

Example:
Input: target = 12, position = [10,8,0,5,3], speed = [2,4,1,1,3]
Output: 3

Constraints:
n == position.length == speed.length
1 <= n <= 105
0 < target <= 106
0 <= position[i] < target
All the values of position are unique.
0 < speed[i] <= 106

Q.8 You are given a large integer represented as an integer array digits, where each digits[i] is the ith digit of the integer. The digits are ordered from most significant to least significant in left-to-right order. The large integer does not contain any leading 0's.
[HINT: Increment the large integer by one and return the resulting array of digits.]

Constraints:

1 <= digits.length <= 100
0 <= digits[i] <= 9
digits does not contain any leading 0's.

Q.9 You are given an integer array deck where deck[i] represents the number written on the ith card.

Partition the cards into one or more groups such that:
Each group has exactly x cards where x > 1, and All the cards in one group have the same integer written on them.

Return true if such partition is possible, or false otherwise.

Example:
Input: deck = [1,2,3,4,4,3,2,1]
Output: true

Constraints:
1 <= deck.length <= 104
0 <= deck[i] < 104

Q.10 Given two integers dividend and divisor, divide two integers without using multiplication, division, and mod operator.

The integer division should truncate toward zero, which means losing its fractional part. For example, 8.345 would be truncated to 8, and -2.7335 would be truncated to -2.

Return the quotient after dividing dividend by divisor.

[Note: Assume we are dealing with an environment that could only store integers within the 32-bit signed integer range: [−231, 231 − 1]. For this problem, if the quotient is strictly greater than 231 - 1, then return 231 - 1, and if the quotient is strictly less than -231, then return -231.]

Example:
Input: dividend = 10, divisor = 3

Output: 3

Constraints:
-231 <= dividend, divisor <= 231 - 1
divisor != 0

Q.11 You are given a 0-indexed string word, consisting of lowercase English letters. You need to select one index and remove the letter at that index from word so that the frequency of every letter present in word is equal.

Return true if it is possible to remove one letter so that the frequency of all letters in word are equal, and false otherwise.

[Note: The frequency of a letter x is the number of times it occurs in the string. You must remove exactly one letter and cannot choose to do nothing.]

Example:
Input: word = "abcc"
Output: true

Constraints:
2 <= word.length <= 100
word consists of lowercase English letters only.
[TEST CASE: aazz]