Katas complejas para ejercitar C#

A continuación les dejo algunas katas que considero interesante trabajar. Si les resultan muy difíciles, pueden tomar otras en alguna de las siguientes páginas o también resolver las mismas Katas que tienen para JS, pero en este lenguaje. Si tienen dificultades con el inglés, es un buen ejercicio el intentar traducirlas con o sin ayuda.

- https://www.w3resource.com/csharp-exercises/basic/index.php
- https://ankitsharmablogs.com/csharp-coding-questions-for-technical-interviews/

Los siguientes cuatro ejercicios son tomados de la página https://www.codility.com/. Allí pueden resolverlos por tiempo y con casos de prueba automatizados.

a) Torneo de tennis

You are hosting a tennis tournament. P players, who will take part in the first round of this tournament, are already registered and you have reserved C tennis courts for the matches. Exactly two players play in each game and only one game can be played on each court at any given time. You want to host the maximum possible number of games starting at the same time (in order to finish the first round quickly).

How many games can be hosted in parallel simultaneously?

Write a function:

class Solution { public int solution(int P, int C); }

that, given the number of players P and the number of reserved courts C, returns the maximum number of games that can be played in parallel.

Examples:

1. Given P = 5 players and C = 3 available courts, the function should return 2. Two games can be played simultaneously (for instance, the first and second players can play on the first court, and the third and fourth players on the second court, and the third court will be empty because the fifth player does not have a partner to play with).

2. Given P = 10 players and C = 3 courts, the function should return 3. At most three games can be hosted in parallel.

b) Contraseña bancaria

You would like to set a password for a bank account. However, there are three restrictions on the format of the password:

- it has to contain only alphanumerical characters (a−z, A−Z, 0−9);
- there should be an even number of letters;
- there should be an odd number of digits.

You are given a string S consisting of N characters. String S can be divided into *words* by splitting it at, and removing, the spaces. The goal is to choose the longest word that is a valid password. You can assume that if there are K spaces in string S then there are exactly K + 1 words.

For example, given "test 5 a0A pass007 ?xy1", there are five words and three of them are valid passwords: "5", "a0A" and "pass007". Thus the longest password is "pass007" and its length is 7. Note that neither "test" nor "?xy1" is a valid password, because "?" is not an alphanumerical character and "test" contains an even number of digits (zero).

Write a function:

class Solution { public int solution(String S); }

that, given a non-empty string S consisting of N characters, returns the length of the longest word from the string that is a valid password. If there is no such word, your function should return −1.

For example, given S = "test 5 a0A pass007 ?xy1", your function should return 7, as explained above.

c) <u>String reversible</u>

Write a function:

class Solution { public int solution(String S); }

that, given a string S, returns the index (counting from 0) of a character such that the part of the string to the left of that character is a reversal of the part of the string to its right. The function should return −1 if no such index exists.

Note: reversing an empty string (i.e. a string whose length is zero) gives an empty string.

For example, given a string:

"racecar"

the function should return 3, because the substring to the left of the character "e" at index 3 is "rac", and the one to the right is "car".

Given a string:

"x"

the function should return 0, because both substrings are empty.

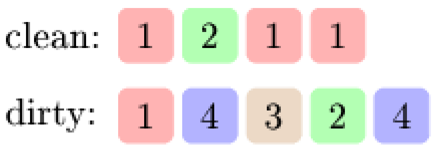
d) <u>Medias sucias</u>

Bob is about to go on a trip. But first he needs to take care of his supply of socks. Each sock has its own color. Bob wants to take as many pairs of clean socks as possible (both socks in the pair should be of the same color).

Socks are divided into two drawers: clean and dirty socks. Bob has time for only one laundry and his washing machine can clean at most K socks. He wants to pick socks for laundering in such a way that after washing he will have a maximal number of clean, same-colored pairs of socks. It is possible that some socks cannot be paired with any other sock, because Bob may have lost some socks over the years.

Bob has exactly N clean and M dirty socks, which are described in arrays C and D, respectively. The colors of the socks are represented as integers (equal numbers representing identical colors).

For example, given four clean socks and five dirty socks:



If Bob's washing machine can clean at most K = 2 socks, then he can take a maximum of three pairs of clean socks. He can wash one red sock and one green sock, numbered 1 and 2 respectively. Then he will have two pairs of red socks and one pair of green socks.

Write a function:

class Solution { public int solution(int K, int[] C, int[] D); }

that, given an integer K (the number of socks that the washing machine can clean), two arrays C and D (containing the color representations of N clean and M dirty socks respectively), returns the maximum number of pairs of socks that Bob can take on the trip.

For example, given K = 2, C = [1, 2, 1, 1] and D = [1, 4, 3, 2, 4], the function should return 3, as explained above.