Guía de ejercicios de repaso

ACLARACION: al menos dos tests unitarios por cada ejercicio y, para entregar esta guía, hay dos opciones:

a. *Se entregan los ejercicios del 1 al 9 con sus test unitarios andando correctamente*
b. *Se entregan los ejercicios del 10 al 12 con sus test unitarios andando correctamente*

1. Dada una lista de palabras de forma singular, retornar un SET de esas palabras en forma plural si aparecen mas de una vez en la lista.

   pluralize(["cow", "pig", "cow", "cow"]) → { "cows", "pig" }
   pluralize(["table", "table", "table"]) → { "tables" }
   pluralize(["chair", "pencil", "arm"]) → { "chair", "pencil", "arm" }

2. Escribir una función que retorne True si cada secuencia consecutiva de unos es seguida por una secuencia consecutiva de ceros de la misma longitud

   same_length("1100011100010") → True
   same_length("101010110") → False
   same_length("111100001100") → True
   same_length("111") → False

3. Crear una función que permita realizar una operación aritmética que incluya adición, sustracción, multiplicación y división teniendo como parámetro de la función los siguientes ejemplos: “12 + 24” , “23 - 21”, “12 // 12”, “12 * 21”
   En el caso de la división por cero, retornar -1 o algún mensaje que diga que no se puede realizar la operación.

   operacion_aritmetica("12 + 12") → 24 // 12 + 12 = 24
   operacion_aritmetica("12 - 12") → 24 // 12 - 12 = 0
   operacion_aritmetica("12 * 12") → 144 // 12 * 12 = 144
   operacion_aritmetica("12 // 0") → -1 // 12 / 0 = -1

4. Crear una función que tome dos listas y determine si existe lo que en poker se denomina “color” (cinco cartas del mismo palo). La primer lista son las 5 cartas repartidas en la mesa. La segunda lista representa 2 cartas en tu mano. Notación: número y palo de la carta (S = Espadas, H = Corazones, D = Diamantes, C = Picas) separados por un guion bajo.
   - check_flush(["A_S", "J_H", "7_D", "8_D", "10_D"], ["J_D", "3_D"]) → True // diamantes
   - check_flush(["10_S", "7_S", "9_H", "4_S", "3_S"], ["K_S", "Q_S"]) → True // espadas
   - check_flush(["3_S", "10_H", "10_D", "10_C", "10_S"], ["3_S", "4_D"]) → False

5. Un número es “Disarium” si la suma de los dígitos elevados a su respectiva posición da como resultado el mismo número en sí mismo. Crear una función que determine si un número es o no “Disarium”.

- is_disarium(75) → False # 7^1 + 5^2 = 7 + 25 = 32
- is_disarium(135) → True # 1^1 + 3^2 + 5^3 = 1 + 9 + 125 = 135
- is_disarium(544) → False
- is_disarium(518) → True
- is_disarium(466) → False
- is_disarium(8) → True

6. Crear una función que eleve al cuadrado cada dígito del número pasado por parámetro.
   - square_digits(9119) → 811181
   - square_digits(2483) → 416649
   - square_digits(3212) → 9414

7. Recibiendo una lista de números, escribir una función que retorne una lista sin números duplicados y ordenados de menor a mayor.
   - unique_sort([1, 2, 4, 3]) → [1, 2, 3, 4]
   - unique_sort([1, 4, 4, 4, 4, 4, 3, 2, 1, 2]) → [1, 2, 3, 4]
   - unique_sort([6, 7, 3, 2, 1]) → [1, 2, 3, 6, 7]

8. Crear una función que tome dos listas y retorne True, si la segunda lista continua el orden de la primer lista y False de lo contrario. Determina si la segunda lista es la primer lista “movida” en una posición hacia la derecha.
   - simon_says([1, 2], [5, 1]) → True
   - simon_says([1, 2], [5, 5]) → False
   - simon_says([1, 2, 3, 4, 5], [0, 1, 2, 3, 4]) → True
   - simon_says([1, 2, 3, 4, 5], [5, 5, 1, 2, 3]) → False

9. Crear una función que retorne la suma de todos los presupuestos de las personas (budget).
   - get_budgets([
     { "name": "John", "age": 21, "budget": 23000 },
     { "name": "Steve", "age": 32, "budget": 40000 },
     { "name": "Martin", "age": 16, "budget": 2700 }
     ]) → 65700

   - get_budgets([
     { "name": "John", "age": 21, "budget": 29000 },
     { "name": "Steve", "age": 32, "budget": 32000 },
     { "name": "Martin", "age": 16, "budget": 1600 }
     ]) → 62600

Ejercicios de algoritmos complejos

10. Crear una función que tome un array de numeros y retorne la palabra “Boom!” si el dígito 7 aparece en el array. Sino, que retorne “No se encuentra el número 7 en el array” .
    Ejemplo:
    - sevenBoom([1, 2, 3, 4, 5, 6, 7]) → "Boom!" // El array contiene el número 7
    - sevenBoom([8, 6, 33, 100]) → “No se encuentra el número 7 en el array” // ninguno de los elementos contiene el número 7

- sevenBoom([2, 55, 60, 97, 86]) ⟞ "Boom!" // 97 contiene el número 7.

11. Escribí una función que reciba dos parámetros: un string S y un integer R. La función debe devolver un string donde los caracteres consecutivos de S no se repitan más que R veces. Tiene que devolver un string con el texto limpio y la cantidad de caracteres repetidos correcta.
    Ejemplos:
    - "AAA", 2 => "AA"
    - "AAAAAFFFFOOOA", 2 => "AAFFOOA"
    - "111223333344", 1 => "1234"
    - "AABB", 1 => "AB"

12. Escribir una función recursiva que reciba un conjunto de caracteres únicos, y un número k, e imprima todas las posibles cadenas de longitud k formadas con los caracteres dados (permitiendo caracteres repetidos).
    Ejemplo:
    - combinaciones(['a', 'b', 'c'], 2) debe imprimir aa ab ac ba bb bc ca cb cc