Boletín 2. Programación modular.

Para cada función debes realizar los tests correspondientes que cubran la mayor cantidad de casos de prueba.

1. Crea un programa que genere 100 números de forma aleatoria y que posteriormente ofrezca al usuario la posibilidad de:
    a. Conocer el mayor.
    b. Conocer el menor.
    c. Obtener la suma de todos los números.
    d. Obtener la media.
    e. Sustituir el valor de un elemento por otro número introducido por teclado.
    f. Mostrar todos los números.

    ⇒ Implementa el código de cada una de las opciones con funciones y sus correspondientes tests.

    ⇒ Utiliza la función randint para generar números aleatorios (entre 0 y 1000).

```
from random import randint

numero = randint(0,1000)
```

2. Realiza un programa que reciba 10 números y devuelva otra lista con estos números desplazados una posición a la derecha, de tal forma que el último pase a la primera posición, el primero a la segunda, el segundo a la tercera, y así sucesivamente.

    Opcional: Añade un parámetro (D/I) a la función para que el controle el sentido del desplazamiento (a derechas/izquierdas) y otro que indique el número de posiciones a desplazar (0: quedaría igual, 1: desplaza una posición, etc.).

3. Diseña una función denominada **es_primo** que reciba un número y determine si el número es primo o no. Un número es primo si es divisible por 1 y el mismo número.

4. Crea una función **obtener_primos_menores** que reciba un número y devuelva los números primos que son menores o iguales que el proporcionado como argumento. NO debes utilizar la función anterior es_primo, sino el algoritmo dado por la criba de Eratóstenes. Por ejemplo, si se proporciona el número 30, se devolverá: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29]

5. Crea un programa que lea por teclado números de forma sucesiva y los guarde en una estructura de datos; el proceso de lectura y guardado finalizará cuando metamos un número negativo. En ese momento se mostrará el mayor elemento introducido y aquellos que son números primos.

6. Realiza una función **reverse** que reciba una lista de elementos y devuelva otra cuyo contenido sea igual a la original pero invertida. Así, dada la lista [‘Di’, ‘buen’, ‘día’, ‘a’, ‘papa’], deberá devolver [‘papa’, ‘a’, ‘día’, ‘buen’, ‘Di’].

7. Diseña una función llamada **esta_ordenada** que reciba una lista de números y un parámetro (creciente, por defecto a True) y devuelva True si está ordenada o False en caso contrario.

8. Escribir una función denominada **encajan** que reciba dos fichas de dominó en formato [n,n], con $0 \leq n \leq 6$, e indique si dos fichas de dominó encajan o no. Las fichas son recibidas en dos cadenas de texto con el siguiente formato

   `[3,4] [2,5]`

9. Elabora cuatro funciones que reciban una lista variable de números y devuelva una matriz de tres filas que contenga:
   a. primera fila: una lista con todos los que sean primos.
   b. segunda fila, primer valor: el sumatorio
   c. segunda fila, tercer valor: el promedio de los valores.
   d. tercera fila: una lista con el factorial de cada uno de los números.

10. Desarrolla un programa que a partir de una lista de números y un entero k, realice la llamada a tres funciones:
    a. para devolver una lista de números con los menores de k
    b. otra con los mayores
    c. otra con aquellos que son múltiplos de k

11. Diseña una función **conversor** que convierta un número de binario a decimal o de decimal a binario. Esta función recibirá un número en formato de cadena de texto cuya última posición indica el sistema numérico utilizado (D-decimal, B-binario).

    Debe validar la información, así, por ejemplo, el número ‘1020101B’ no sería válido puesto que los valores en binario son 0 y 1. Se recomienda realizar funciones auxiliares (por ejemplo, decidir tipo de dato recibido, convertir a binario, convertir a decimal, etc.).

12. Escribe una función **intersect** que reciba dos listas y devuelva otra lista con los elementos que son comunes a ambas, sin repetir ninguno. Las estructuras de datos originales no deben verse modificadas.

13. Escribe una función **union** que reciba dos listas y devuelva los elementos que pertenecen a una, o bien, a la otra, pero sin repetir ninguno (unión de conjuntos). Las estructuras de datos originales no deben verse modificadas.

14. Escribe una función que, dada una lista de nombres y una letra, devuelva una lista con todos los nombres que empiezan por dicha letra. Debe validar la información.

15. Escribe una función que, dada una lista de cadenas, devuelva la cadena más larga. Si dos o más cadenas miden lo mismo y son las más largas, la función devolverá la que tenga el mayor número de caracteres repetidos .