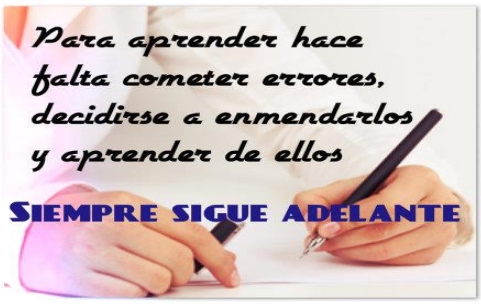
ELEMENTOS DE COMPUTACIÓN – PRUEBA CORTA 3 – ASINCRÓNICA

INSTRUCCIONES:

- Parte sincrónica (40 minutos): explicación de la evaluación y consultas. Pueden hacer la evaluación los que estén presentes en esta parte.
- Parte asincrónica: desarrollo de evaluación. Entrega: 16 de mayo 11pm.
- Desarrollar en Python 3 según material estudiado a la fecha.
- Ponga todas las funciones en un solo programa fuente con el nombre **prueba_corta3_su_nombre.py**. Cada ejercicio debe aparecer con su número en el orden en que está en la evaluación y con el nombre indicado. Puede usar funciones adicionales.
- Subir solución al tecDigital EVALUACIONES / PRUEBA CORTA 3. PARA TODO TRABAJO ENVÍADO POR MEDIOS DIGITALES DEBE ASEGURARSE QUE ENVIÓ EL TRABAJO RESPECTIVO AL LUGAR INDICADO.
- Requisito para revisar la evaluación: las funciones deben tener documentación interna (COMENTARIOS EN EL PROGRAMA FUENTE): al menos lo que hace la función, entradas, salidas.
- El estudiante debe ser el autor de la evaluación enviada. El plagio (copiar en lo sustancial obras ajenas dándolas como propias -RAE, 2021-) en cualquier evaluación conlleva la aplicación del reglamento institucional respectivo, artículo 75.
- Se coordinará día y hora para revisar la evaluación junto con el estudiante el cual mostrará el dominio de la solución implementada desde el punto de vista técnico (uso de conceptos de programación y del lenguaje) así como de la funcionalidad (lo que hace la solución). La revisión puede constar de las siguientes actividades:
    - Revisar esta solución particular
    - Revisar conceptos incluidos en la evaluación
    - Aplicar otras actividades con una complejidad igual o menor a la evaluación.

RECOMENDACIÓN. Siga la metodología de solución de problemas: entender el problema, diseñar un algoritmo, codificar y probar programa. En la etapa de diseño del algoritmo haga un esquema para determinar el comportamiento o patrón del algoritmo para luego proceder con su desarrollo.

NO SE VAYA DIRECTAMENTE A PROGRAMAR EN LA COMPUTADORA, PRIMERO PLANIFIQUE LA SOLUCIÓN HACIENDO ESTE ESQUEMA SUGERIDO.



ELEMENTOS DE COMPUTACIÓN – PRUEBA CORTA 3 – ASINCRÓNICA

EJERCICIO 1

En matemáticas un número abundante n cumple la siguiente propiedad: la suma de los divisores positivos de *n* incluyendo al propio *n* es mayor al producto de *2 * n*. Ejemplos:

El número 12 es un número abundante: Divisores: 1, 2, 3, 4, 6, 12
Suma de los divisores: 28
Fórmula (2 * n): 24

El número 16 no es un número abundante: Divisores: 1, 2, 4, 8, 16
Suma de los divisores: 31
Fórmula (2 * n): 32

Desarrolle la función **numeros_abundantes_en_rango** que reciba un rango de dos números y retorne la lista de números abundantes encontrados en ese rango (8P). La lista tendrá por cada número abundante encontrado una lista donde el primer elemento es el número abundante (15P) y los siguientes elementos son sus divisores (25P). Validar que los dos valores de entrada sean enteros (1P) y que el primer número sea <= que el segundo (1P), de lo contrario retornar una lista vacía. Considere que el primer número abundante que puede encontrar es el 12.

Ejemplos del funcionamiento:

```
>>> numeros_abundantes_en_rango(-5, 19)
[ [12, 1, 2, 3, 4, 6, 12], [18, 1, 2, 3, 6, 9, 18] ]

>>> numeros_abundantes_en_rango(10.51, 15)
[ ]
```

ELEMENTOS DE COMPUTACIÓN – PRUEBA CORTA 3 – ASINCRÓNICA

EJERCICIO 2

Desarrollar la función **validar_contraseña**. Recibe un string que representa una contraseña y retorna el código 0 si la contraseña es correcta (6P). En caso de que la contraseña sea incorrecta retorna un código dependiendo del tipo de error. Con el primer error encontrado la función termina. Validar las restricciones del dato de entrada (4P) y si hay error retornar el código de error 9.

Una contraseña es correcta cuando cumple con todas las condiciones siguientes:

- Tiene al menos 2 caracteres, iguales o diferentes de categoría 1: ABCDEFGHIJKLMNÑOPQRSTUVWXYZ (4P). Un error aquí retorna el código de error 1 (2P).
- Tiene al menos 2 caracteres, iguales o diferentes de categoría 2: abcdefghijklmnñopqrstuvwxyz (4P). Un error aquí retorna el código de error 2 (2P).
- Tiene al menos 2 caracteres, iguales o diferentes de categoría 3: 0123456789 (4P). Un error aquí retorna el código de error 3 (2P).
- Tiene al menos 2 caracteres, iguales o diferentes de categoría 4: carácter diferente a las categorías anteriores (8P). Un error aquí retorna el código de error 4 (2P).
- Caracteres de la misma categoría no pueden estar juntos (10P). Un error aquí retorna el código de error 5 (2P).

Ejemplos del funcionamiento:

```
>>> validar_contraseña(“hOjA*8+5”)
0               # Contraseña correcta

>>> validar_contraseña(“hC9-Ch*9C”)
0               # Contraseña correcta

>>> validar_contraseña(“hMt4*8+52”)
1               # De categoría 1 solo tiene 1 caracter

>>> validar_contraseña(“hM4T*8+52”)
2               # De categoría 2 solo tiene 1 caracter

>>> validar_contraseña(“JhMt*8+”)
3               # De categoría 3 solo tiene 1 carácter

>>> validar_contraseña(“hR5BjA8B+”)
4               # De categoría 4 solo tiene 1 caracter

>>> validar_contraseña(“hOjA*85+”)
5               # Caracteres de la misma categoría no pueden estar juntos

>>> validar_contraseña(12345678)
9               # La entrada debe ser un string
```