Esta es la Tarea 1 del curso *Estructuras de Datos*, 2023-2. La actividad se debe realizar de forma **individual**. Sus soluciones deben ser entregadas a través de Discord a más tardar el día **6 de Agosto a las 22:00**. En caso de dudas y aclaraciones puede escribir por el canal #`tareas` en el servidor de *Discord* del curso o comunicarse directamente con el profesor y/o el monitor.

Condiciones Generales

- Para la creación de su código y documentación del mismo use nombres en lo posible cortos y con un significado claro. La primera letra debe ser minúscula, si son más de 2 palabras se pone la primera letra de la primera palabra en minúscula y las iniciales de las demás palabras en mayúsculas. Además, para las operaciones, el nombre debe comenzar por un verbo en infinitivo. Esta notación se llama *lowerCamelCase*.

  Ejemplos de funciones: `quitarBoton`, `calcularCredito`, `sumarNumeros`

  Ejemplos de variables: `sumaGeneralSalario`, `promedio`, `nroHabitantes`

- Todos los puntos de la tarea deben ser realizados en un único archivo llamado `tarea1.py`. Debe incluir una cabecera con su nombre y código y comentarios que permitan distinguir a qué ejercicio corresponde cada operación en el archivo.

- En los ejercicios en los que se solicita leer datos de la forma en la que se indica en el enunciado y las instrucciones `input` no deben contener ningún mensaje. Así mismo, no se deben agregar mensajes en los que se indique los datos que se deben digitar. De esta manera, se puede asumir que el usuario conoce los datos que debe digitar y que no cometerá errores durante el proceso.

Ejercicios

1. **[9 pts.]** Escriba la definición de la operación `split` que recibe una cadena de texto `cad` y una cadena de un solo caracter `sep` y produce una lista que contiene todas las cadenas que resultan de dividir la cadena `cad` con respecto a la cadena `sep`. A partir de lo anterior, una invocación como la siguiente:

   ```
   split("hola mundo cruel!", " ")
   ```

   debe producir como resultado la lista:

   ```
   ["hola", "mundo", "cruel!"]
   ```

   En tanto que la invocación:

   ```
   split("hola mundo bonito!", "o")
   ```

   debe producir como resultado la lista:

   ```
   ["h", "la mund", " b", "nit", "!"]
   ```

   Nota: No está permitido usar en su solución la operación `split` nativa de Python.

2. **[9 pts.]** Escriba la definición de la operación `obtenerSumas` que recibe una matriz `mat` de dimensiones $N \times N$ y produce una lista que tiene $2 \cdot N - 1$ posiciones y en cada posición i tiene la suma de los valores en la matriz `mat` cuyos índices suman i. A partir de lo anterior, si se tiene una matriz `mat` como sigue:

```
mat = [[10, 6, 5],
       [14, 30, 92],
       [12, 45, 58]]
```

y se realiza la invocación `obtenerSumas(mat)` se debería producir como resultado la lista:

```
[10, 20, 47, 137, 58]
```

Es importante resaltar que:

- El valor 10 está en la posición 0 y corresponde a la suma de las posiciones de la matriz cuyos índices suman 0: `10 = mat[0][0]`.
- El valor 20 está en la posición 1 y corresponde a la suma de las posiciones de la matriz cuyos índices suman 1: `20 = mat[0][1] + mat[1][0]`.
- El valor 47 está en la posición 2 y corresponde a la suma de las posiciones de la matriz cuyos índices suman 2: `47 = mat[2][0] + mat[1][1] + mat[0][2]`.
- El valor 137 está en la posición 3 y corresponde a la suma de las posiciones de la matriz cuyos índices suman 3: `137 = mat[2][1] + mat[1][2]`.
- El valor 58 está en la posición 4 y corresponde a la suma de las posiciones de la matriz cuyos índices suman 4: `10 = mat[2][2]`.

3. **[9 pts.]** Escriba la definición de la operación `obtenerMayorPosicion` que recibe dos cadenas de texto `cad1` y `cad2` y produce como resultado un diccionario en el que las claves son los caracteres de la cadena `cad2` y los valores son las posiciones de la última ocurrencia de dicho caracter en la cadena `cad1`. Si un caracter no aparece en la cadena `cad1` entonces debe estar asociado en el diccionario al valor -1. A partir de lo anterior una invocación como la siguiente:

```
obtenerMayorPosicion("mi corazon encantado vibra por el polvo de esperanza y
    magia!!",
                     "za!em iolpx")
```

debería producir como resultado el siguiente diccionario:

```
{"o" : 38, "z" : 50, "a" : 59, "e" : 46, " " : 54, "m" : 55, "x" : -1, "!" :
    61, "p" : 45, "l" : 36, "i" : 58}
```

4. Desarrolle las operaciones descritas a continuación:

   (a) **[9 pts.]** Escriba la definición de la operación `obtenerDosMayores` que recibe una lista `l` de números y produce como resultado una tupla que contiene los dos números más grandes en `l`. A partir de lo anterior, una invocación como la siguiente:

   ```
   obtenerDosMayores([40, 10, 22, 12, 33, 33, 33, 39, 54])
   ```

   debe producir como resultado la tupla `(54, 40)`.

(b) **[5 pts.]** Escriba la definición de una operación de lectura e impresión de datos en la que primero se lea un número correspondiente a la cantidad de ejecuciones a realizar. Posteriormente, para cada ejecución se deben leer dos líneas que contendrán una cantidad arbitraria de números y que corresponden a los valores que deben estar en la lista con la que se debe ejecutar la función `obtenerDosMayores`. Para cada ejecución se debe imprimir los dos valores obtenidos en la invocación con el formato mostrado en el siguiente ejemplo:

```
2
5 6 10 9 8 50 34
11 65 49
1 2 4 65 71 18
42 50 60 10 20 30 15 23
```

En este ejemplo, se tienen que se realizarán dos ejecuciones (el número en la primera línea). Para la primera ejecución, en la segunda y tercera línea se tienen los elementos que contendrá la lista (los números 5, 6, 10, 9, 8, 50, 34, 11, 65, 49). En tanto que para la segunda ejecución la lista tendrá los valores en la cuarta y quinta línea (los números 1, 2, 4, 65, 71, 18, 42, 50, 60, 10, 20, 30, 15, 23).

Los datos que deben imprimirse para la anterior entrada son los siguientes:

```
Primer valor mayor --> 65, Segundo valor mayor --> 50
Primer valor mayor --> 71, Segundo valor mayor --> 65
```

5. **[9 pts.]** Escriba la definición de la operación `mostrarDivisores` que reciba como parámetro un valor `N` y muestre los divisores de `N`. En adición a esto, esta operación debe también mostrar la suma de dichos divisores. Los resultados deben ser imprimidos siguiendo el formato que se indica a continuación.

   Si se realiza la invovación `mostrarDivisores(100)` se debe imprimir:

```
Divisores número 100:
 --> 1,
 --> 2,
 --> 4,
 --> 5,
 --> 10,
 --> 20,
 --> 25,
 --> 50,
 --> 100

Suma de los divisores del número 100: 217
```