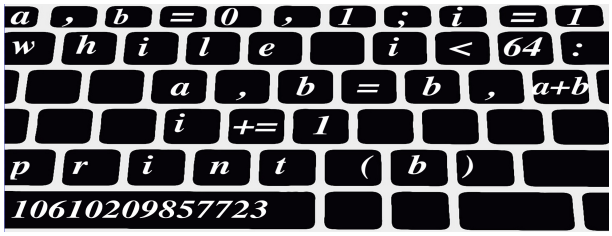

FUNDAMENTOS DE COMPUTACIÓN
GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES
Universidad de Cantabria

Examen Parcial(parte ordenador)
21 de Noviembre del 2020

NOMBRE Y APELLIDOS:

- Crea un archivo por cada ejercicio *ejercicio1.py*, *ejercicio2.py* y *ejercicio3.py*.
- Al finalizar la prueba, subir los 3 archivos por separado al subdirectorio EXAMEN 21/11/2020. del directorio PRUEBAS
- Tener **SOLAMENTE** abiertas en el ordenador las ventanas *Idle* y *shell* de PYTHON.

1. [30 %] Los elementos que están por arriba de la diagonal de la matriz cuadrada 3×3 : $A = \begin{pmatrix} 8 & 2 & 5 \\ -1 & 3 & -1 \\ 5 & -7 & 1 \end{pmatrix}$ son: $\begin{pmatrix} * & 2 & 5 \\ + & * & -1 \\ + & + & * \end{pmatrix}$, y la suma de esos elementos es $6 = 2 + 5 - 1$.

   Y, los elementos que están por arriba de la diagonal de la matriz cuadrada 4×4 : $B = \begin{pmatrix} 2 & 1 & -1 & 0 \\ 0 & -3 & 7 & 5 \\ 1 & 1 & 2 & -1 \\ -5 & 2 & -3 & 3 \end{pmatrix}$ son $\begin{pmatrix} * & 1 & -1 & 0 \\ + & * & 7 & 5 \\ + & + & * & -1 \\ + & + & + & * \end{pmatrix}$ y la suma de esos elementos es 11.

   En PYTHON podemos representar una matriz cuadrada de n filas por n columnas, como una lista de n elementos, formados por listas de n objetos numéricos.

```
>>> A=[[8, 2, 5], [-1, 3, -1], [5, -7, 1] ]
>>> for i in A: i

[8, 2, 5]
[-1, 3, -1]
[5, -7, 1]
>>> B = [ [2, 1, -1, 0], [0, -3, 7, 5 ], [1, 1, 2, -1], [-5, 2, -3, 3 ] ]
>>> B[1][2]
7
>>>
```

   Se pide, implementar la función `suma_diagonal_superior` que tenga como argumento una matriz cuadrada y devuelva la suma de los elementos por arriba de la diagonal:

```
>>> suma_diagonal_superior([ [8, 2, 5], [-1, 3, -1], [5, -7, 1]  ] )
6
>>> suma_diagonal_superior(B)
11
>>> suma_diagonal_superior([ [1, 2], [3, 4] ] )
2
>>> suma_diagonal_superior([ [3, 4], [1, 2] ] )
4
```

2. [30 %] Escribir la función `invertir`, que tenga como argumento una cadena(`str`) o una tupla(`tuple`) o una lista(`list`) y devuelva una cadena/tupla/lista, respectivamente, con sus elementos invertidos.

```
>>> invertir('abcb1d')
'd1bcba'
>>> invertir( (1, -2,  [3.5, 4], 'hola' ) )
('hola', [3.5, 4],  -2, 1)
>>> invertir( [1, 2.3, (5,'b'), 1] )
[1, (5,'b'),  2.3, 1]
```

3. [40 %] Se pide diseñar una función `juntar_ordenando` que tenga como argumento dos listas con datos numéricos, ordenadas de mayor a menor y, devuelva una única lista con todos sus elementos ordenados de menor a mayor. Para ello deberéis implementar la siguiente estrategia. Supongamos que tenemos dos pilas de cartas (las dos listas) con la cara por arriba. Cada pila está ordenada tal que la carta más grande es la superior. Queremos mezclar las dos pilas para obtener una única pila ordenada, de menor a mayor.

   - Elegimos la más grande entre de las dos cartas superiores de cada pila;
   - la quitamos de esa pila y la ponemos en una nueva pila con la cara por abajo;
   - repetimos este procedimiento hasta que no haya cartas en alguna de las pilas;
   - entonces sólo tenemos que **invertir** el resto de las cartas de la otra pila y ponerlas boca abajo en la nueva pila.

```
>>> juntar_ordenando([5,5,4,1,1,0], [6,3,1,0,-1])
[-1, 0, 0, 1, 1, 1, 3, 4, 5, 5, 6]
>>> juntar_ordenando([1, 0], [3 ,2, -1])
[-1, 0, 1, 2, 3]
>>> juntar_ordenando([7, 0], [9, 2])
[0, 2, 7, 9]
>>>
```