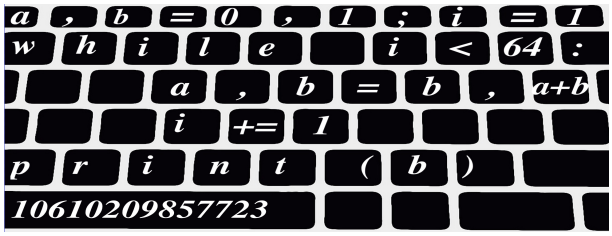

FUNDAMENTOS DE COMPUTACIÓN
GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES
Universidad de Cantabria

Examen Extraordinario (parte ordenador)
15 de febrero del 2021

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 15/2/2021 del directorio PRUEBAS
- En el ordenador tener abiertas **SOLAMENTE** las ventanas *Idle* y *shell* de PYTHON.

1. ([30 %]) Elaborar un programa que gestione los datos de coronavirus en Cantabria por municipios de un día (Servicio Cantabro de Salud). Descargar de Moodle el archivo `Covid19municipalizado.txt` con los datos: Código Municipio, Nombre Municipio, Activos, Curados, Casos y Fallecidos.

```
39001,ALFOZ DE LLOREDO,1,65,68,2
39002,AMPUERO,5,107,117,5
39003,ANIEVAS,0,17,17,0
...
39101,VILLAVERDE DE TRUCIOS,0,1,1,0
39102,VOTO,8,42,52,2
39999,DESCONOCIDO,75,630,708,3
```

   Se pide:

   - Almacenar la información del archivo leído en un diccionario: la CLAVE de los elementos del diccionario será el código del Municipio y, el VALOR una lista con el resto de la información.
   - Presentar un menú con las opciones siguientes:
     1. Listar la información del archivo. Comprobar que los datos del archivo son correctos, es decir, que casos = activos +curados + fallecidos.
     2. Buscar y mostrar información de un municipio.
     3. Modificar los datos Covid-19 de un municipio, esto es: activos, curados, casos y/o fallecidos.
     4. Salir y escribir la información del diccionario en un archivo de texto.

   Usar una función para cada una de las opciones 2 y 3 del menú y para el propio menú.

2. ([30 %]) **Quién da nombre a esta sucesión?**— El título de este problema incurre en una flagrante falta de ortografía, pues es sabido que, a diferencia de otros idiomas, en el nuestro les ocurre a los signos de interrogación —y a los de admiración— como a los paréntesis, corchetes y llaves en matemáticas: que van por parejas.

   En una expresión matemática no solo encontramos el mismo número de paréntesis de apertura que de cierre. Además, aparecen «bien anidados». Por ejemplo, no podemos encontrar el patrón ())(, pero sí ()() o (()). Diseña un programa que devuelva todos los patrones válidos con un número arbitrario de pares de paréntesis.

```
>>> parentesis(1):
['()']
>>> parantesis(2)
['(())', '()()']
>>> parentesis(3)
['((()))', '(())()', '(()())', '()()()']
>>> parentesis(4)
['(((())))', '((()))()', '((())())', '(())()()', '((()()))', '(()())()', '(()()())', '()()()()']
>>> parentesis(5)
['((((()))))', '(((()))())', '(((())()))', '((()))()()', '((()()()))', '((()))()()', '((()))()())',
'(())()()()', '((((()))))', '(((())))()', '(((()())))', '((())()()', '(((()())))', '((()()))()',
 '((()()()))', '()()()()()']
```

3. ([40 %]) En computación un CONJUNTO es una estructura de datos que representa los conjuntos finitos, que ya conocemos en matemáticas y los cuáles son una colección de diferentes valores sin ningún orden alguno ni valores duplicados. Una lista no es un conjunto porque es posible que un elemento aparezca repetidas veces. Podemos simular el tipo de datos `conjunto` con una simple lista, pero deberíamos tener presente muchos aspectos. Por ejemplo, precaución de no insertar un elemento si ya está presente, además las listas tienen orden. Los diccionarios no están ordenados, pero sus elementos están asociados a las claves. Para no complicar nuestros programas, es mejor definir un nuevo tipo de datos que, internamente, realice las comprobaciones pertinentes: la clase **Conjunto**.

```
class Conjunto(object):
    def __init__(self, lista=[]):
        self.elementos =lista
```

   1. Definir el método `__eq__` para que devuelva el booleano TRUE si los dos objetos son iguales o FALSE, en caso contrario
   2. Definir el método `__str__` para mostrar, apareciendo por pantalla con sus elementos separados por comas y encerrados entre llaves.
   3. Implementar un método `inserta` que con dos argumentos: `self`(como siempre) y el elemento que queremos añadir, pero antes debemos comprobar que no está presente.
   4. Implementar un método `borrar` que con dos argumentos: `self` y el elemento que queremos eliminar, obviamente solo habrá que borrar el elemento de la lista si pertenece.
   5. Implementar el método `cardinal` para que devuelva el número de elementos del objeto conjunto.
   6. Definir los métodos `union` e `interseccion` con dos argumentos, `self` y `otro` y devuelva el objeto unión e intersección.
   7. Adapta el método especial `__sub__(self`otro), para devolver el objeto de los elementos que están `self` pero no están en `otro`.

Ejemplo de ejecución del programa

```
>>> A = Conjunto([2,1,3])
>>> B = Conjunto([1,3,2,1])
>>> C= Conjunto([7,3,6,1,3])
>>> A == B
True
>>> print(B)
{1,3,2}
>>> A.inserta(5)
>>> A.inserta(2)
>>>print(A)
{2,1,3,5}
>>> print(B.borrar(3))
{1,2}
>>> print(A.union(C))
{2,1,3,5,7,6}
>>> print(A.interseccion(C))
{1,3}
>>> C.cardinal()
4
>>> print(C - A)
{7,6}
```