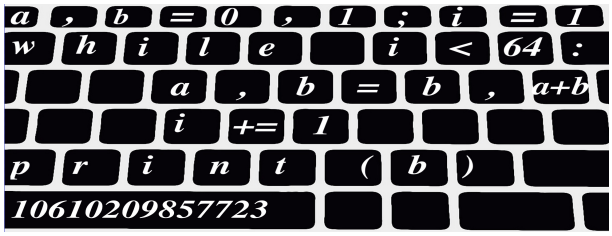


FUNDAMENTOS DE COMPUTACIÓN
Grado en ingeniería en tecnologías Industriales
Universidad de Cantabria

Examen Extraordinario (parte escrita)
7 de febrero del 2022

Nombre y apellidos:

---

1. [15 %] Se consideran las siguientes dos funciones no documentadas.

h.py

```
def h(s, d):
    for k in d.keys():
        d[k] = 0
    for c in s:
        if c in d:
            d[c] += 1
        else:
            d[c] = 0
    return d
```

agra.py

```
def agra(d):
    resultado = 0
    for k in d:
        resultado += d[k]
    return resultado
```

¿Qué se muestra por pantalla después de ejecutar estos dos programas?

```
>>> d1 = {}; d2 = d1; d1 = h('abbc', d1)
>>> print(agra(d1))
```

<---------- SOLUCION

```
>>> d2 =h('bbcaa', d2)
>>> print(agra(d2))
```

<---------- SOLUCION

```
>>> print(h(' ',{}))
```

<---------- SOLUCION

```
>>> print(resultado)
```

<---------- SOLUCION

2. [10 %] Se consideran las siguientes funciones mutuamente recursivas:

coger.py

```
def coger(q):
    if len(q) > 0:
        return [q[0]] + saltar(q[1:])
    else:
        return []
```

saltar.py

```
def saltar(q):
    if len(q)<1:
        return []
    return coger(q[1:])
```

¿Cuál es la salida de `coger`$([3,7,1])$ y `saltar`$([3,7,1])$?

`coger([3,7,1])`	`saltar([3,7,1])`

3. [15 %] Se consideran las siguientes dos clases no documentadas.

A.py

```
class A(object):
    x = 5
    y = 10

    def __init__(self,x):
        self.z = x

    def f(self,x):
        if x > 0:
            return 2+self.f(x-1)
        else:
            return x +self.x
```

B(A).py

```
class B(A):
    x = 3

    def __init__(self, x):
        self.w = self.f(x-3)

    def g(self, x):
        if x > 0:
            return self.f(x-1)
        else:
            return x + self.y
```

¿Qué resultará de ejecutar las siguientes sentencias?

```
>>> z = B(4)
>>> y = A(5)
>>> z.f(3)
```

<-------------- SOLUCIÓN

```
>>> z.g(2)
```

<-------------- SOLUCIÓN

```
>>> y.f(6)
```

<-------------- SOLUCIÓN

```
>>> y.g(12)
```

<-------------- SOLUCIÓN

4. [15 %] Responder brevemente a las siguientes cuestiones:

   - ¿Para qué sirve el método `commit` de sqlite3?

   - ¿Para qué sirve la aplicación DB Browser?

   - ¿Deben ser los valores de un diccionario objetos inmutables?

   - ¿Cuáles son las componentes que constituyen la CPU(Central Processing Unit) de un ordenador?

   - Dados los números en hexadecimal M=AC70.3B y N= 6A4.0E5, se pide en binario

     `M + N:` `M - N:`

   - ¿Qué es el Universal Character Set(UCS)?

5. [10 %] ¿Qué se muestra por pantalla después de ejecutar el siguiente programa?

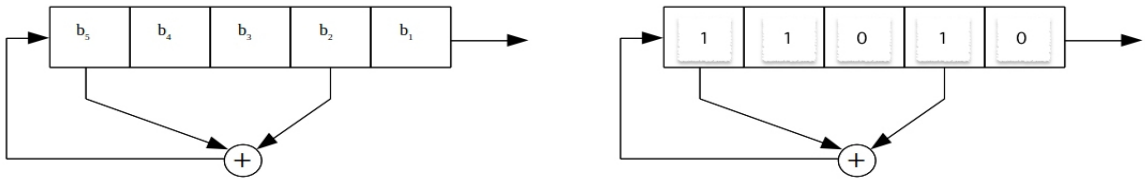
```
T = (0.1, 0.1)
x = 0.0
i = 0
while i < len(T):
    for j in T:
        x += i +j
        print(x)
    i += 1
print(i)
```

<---------- SOLUCIÓN

6. [15 %] Escribe la función `es_tupla_enteros_ordenada(valor)` teniendo como argumento `valor` cualquier objeto Python; devolviendo `True` si `valor` es una tupla, no vacía y ordenada de enteros, y `False` en otro caso.

```
>>> es_tupla_enteros_ordenada((1,2,1))
False
>>> es_tupla_enteros_ordenada((1,2,2))
True
>>> es_tupla_enteros_ordenada([1,2,2])
False
>>> es_tupla_enteros_ordenada((1.1,2,4))
False
```

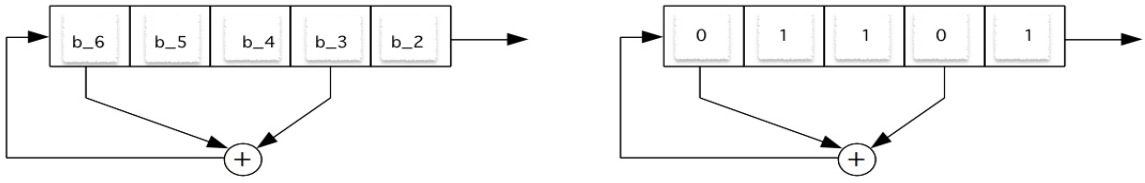
7. [20 %] LFSR son las iniciales de Linear Feedback Shift Register, que se traduce como: registro de desplazamiento con retroalimentación lineal. Es un registro de desplazamiento en el cual la entrada es un bit proveniente de aplicar una función de transformación lineal a un estado anterior. Ilustramos el procedimiento con el siguiente ejemplo. Partimos de una semilla $b_1b_2b_3b_4b_5 = 01011$ y de la transformación lineal fija $c_1c_2c_3c_4c_5 = 01001$.



El bit b_6 que produce el LFSR se obtiene como:

$$b_1 \times c_1 + b_2 \times c_2 + b_3 \times c_3 + b_4 \times c_4 + b_5 \times c_5 = 0 \times 0 \quad + 1 \times 1 \quad + \quad 0 \times 0 \quad + \quad 1 \times 0 \quad + \quad 1 \times 1 = 2$$

Y reducimos modulo 2, es decir `2 % 2` luego el nuevo bit $b_6 = 0$. Para construir otro bit, se retroalimenta el LFSR tomando como nueva semilla $b_2b_3b_4b_5b_6 = 10110$.



y aplicamos el procedimiento anterior, obteniendo: $1 \times 0 + 0 \times 1 + 1 \times 0 + 1 \times 0 + 0 \times 1 = 0$, y $0\,\%2 = 0 = b_7$. El bit b_8 para la semilla $b_3b_4b_5b_6b_7 = 01100$ es $b_8 = 1$, el bit b_9 para la semilla $b_4b_5b_6b_7b_8 = 11001$ es $b_9 = 0$. Y así sucesivamente.

Varios sistemas de comunicación digital usan LFSR, como Posicionamiento Global(GPS), NICAM (digital audio system for television), sistemas criptográficos, etc.

Se pide escribir una función `lfsr` que tenga como argumentos: dos cadenas binarias `s` y `c` de la misma longitud y un entero $n \geq 0$, devolviendo la cadena de los n bits generados por el LFSR que tiene como semilla la cadena `s` y como transformación lineal `c`.

```
>>> lfsr('01011','01001',0)
''
>>> lfsr('01011','01001',1)
'0'
```

```
>>> lfsr('01011','01001',3)
'001'
>>> lfsr('01011','01001',8)
'00100011'
```