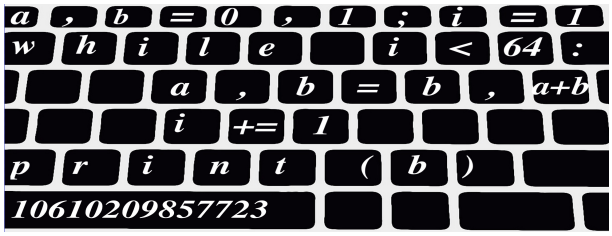


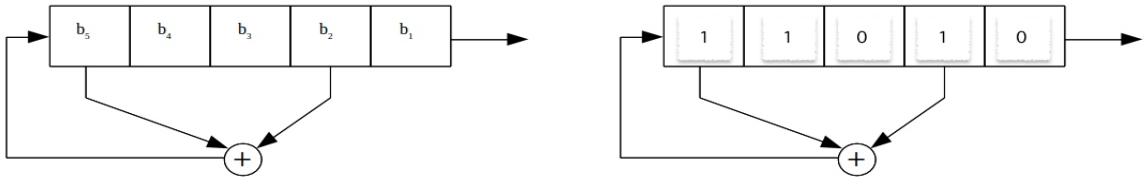
FUNDAMENTOS DE COMPUTACIÓN
GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES
Universidad de Cantabria

Examen Extraordinario (parte ordenador)
7 de febrero del 2022

NOMBRE Y APELLIDOS:

- Crea un archivo por cada ejercicio *ejercicio1.py* y *ejercicio2.py*.
- Al finalizar la prueba, subir los 2 archivos por separado al subdirectorio EXAMEN 7/2/2022 del directorio PRUEBAS
- En el ordenador tener abiertas **SOLAMENTE** las ventanas *Idle* y *shell* de PYTHON.

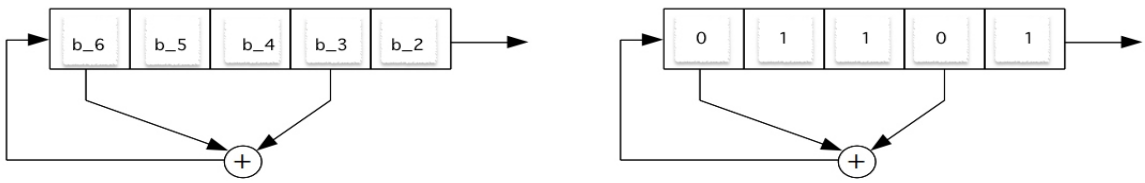
1. [85 %] LFSR son las iniciales de Linear Feedback Shift Register, que se traduce como: registro de desplazamiento con retroalimentación lineal. Es un registro de desplazamiento en el cual la entrada es un bit proveniente de aplicar una función de transformación lineal a un estado anterior. Ilustramos 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'
```

- La secuencia generada por un LFSR tiene un periodo de repetición, es decir, la secuencia vuelve a generarse y se repite indefinidamente.

  ```
  >>> lfsr('01011','01001',32)
  '00100011110101100100011110101 1001'
  ```

  0010001111**01011**0010001111**01011**001.... Se dice que LFSR genera la secuencia `0010001111` de periodo 15, puesto que el número de bits generados es 10 más los 5 bits de la semilla.

  Escribir la función `lfsr_periodo` que tenga como argumentos dos cadenas binarias `s` y `c` de la misma longitud y devuelva el periodo y la cadena de bits que genera.

  ```
  >>> lfsr_periodo('01011','01001')
  ('0010001111', 15)
  >>> lfsr_periodo('1001','0110')
  ('011', 7)
  ```

- Construir la clase `LFSR` que tenga como único atributo la cadena de bits `c` de la transformación lineal.
  1. Implementar el método especial `__str_(self)` para mostrar por pantalla la cadena de bits del atributo del objeto `self`.
  2. Escribir el método `bit_lfsr(self, semilla)` que tenga como argumentos el objeto `self` y una cadena de bits de la misma longitud, y devuelva el bit generado por el LFSR que tiene como transformación lineal el atributo del objeto y como semilla el argumento `semilla`.
  3. Escribir el método `periodo_lfsr(self, semilla)` que tenga como argumentos el objeto `self` y una cadena de bits de la misma longitud y devuelva el periodo y la cadena de bits que el LFSR obtiene con transformación lineal el atributo del objeto y como semilla el otro argumento `semilla`.
  4. Implementar el método especial sumar `__add_(self,otro)` donde las cadenas de bit de los dos objetos tienen la misma longitud, y donde la suma es la $XOR = \bigoplus$ de los bits, es decir

  $$1 \bigoplus 1 = 0, \quad 1 \bigoplus 0 = 1, \quad 0 \bigoplus 1 = 1, \quad 0 \bigoplus 0 = 0$$

  ```
  >>> R = LFSR('01001')
  >>> S = LFSR('11101')
  >>> print(S)
  '11101'
  >>> R.bit_lfsr('01011')
  '0'
  >>> R.periodo_lfsr('01011')
  ('0010001111', 15)
  >>> print(R + S)
  '10100'
  ```

2. [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(())
False
>>> 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
```