Módulo Profesional: Big Data Aplicado

Scala

Ejercicio 1: Calcular si un número es divisible por otro

```
/* Ejercicio 1 - Calcular si un número es divisible por otro
  En esta función el primer parámetros se pasa por llamada por valor y el segundo parámetro (divisor)
  se pasa por llamada por nombre.
   - Si el divisor es igual 0 entonces muestra el siguiente mensaje:
      "El divisor no puede ser 0"
   - Si el divisor es mayor 0 entonces muestra el siguiente mensaje:
      "Calculando si número es divisible por el divisor"
      **número es el valor del parámetro y divisor es el valor del parámetro
*/
```

Ejercicio 2: Calcular el total de un pedido con impuestos y envío

```
/*Ejercicio 2 - Calcular el total de un pedido con impuestos y envío
La función que calcula el total de un pedido en línea, donde la función recibe como parámetros:
  - Subtotal: Valor total de los productos
  - Tasa de impuesto: El porcentaje de impuestos a pagar
  - Envio: El coste de envío a pagar

  Una vez realizado el cálculo la función debe devolver un mensaje que muestre la siguiente información
  con los valores formateados con dos decimales.
    - Precio total
    - Subtotal
    - Tasa de impuestos
    - Envio
*/
```

Ejercicio 3: Buscar un elemento en una lista

```
/*Ejercicio 3 - Buscar un elemento en una lista
Implementa una función que busque si el número existe en la lista de números,
 devolviendo verdadero o falso según sea el caso.
 - Parámetros de la función serán los siguientes:
    List [Int] --> lista de enteros
    numero de tipo Int

   *** Pista:
       Lista.head --> obtenemos el primer elemento de la lista
       Lista.tail --> es una nueva lista que contiene todos los elementos de la lista original excepto el primero
 */
```

Ejercicio 4: Invertir una lista de enteros

```
/*Ejercicio 4 - Inversión de una lista
Escribe una función recursiva que reciba una lista y devuelva la misma lista pero invertida.
Parámetros:
 - Lista de enteros. Lista[Int]

  *** Pista:
        - Nil --> es una lista vacía y es una construcción esencial para trabaja en Scala.
        Se utiliza para terminar la recursión de una lista y sirve como marcador de final de una lista
        - lista.last --> obtener el último elemento de la lista o lanza una excecpión si la lista está vacía.
        - lista.init --> para obtener todos los elementos de una lista excepto el último.


 */
```

Ejercicio 5: Contar la cantidad de veces que aparece una carácter en una cadena

```
/* Ejercicio 5 - Contar la cantidad de veces que aparece un carácter en una cadena
 Escribe una función recursiva que reciba una cadena de texto y un carácter y devuelva el número de veces que el
 carácter aparece en la cadena.
 Parámetros de entrada:
    - Cadena de texto String y un carácter Char.

*/
```