Práctica 8: Introducción a Scala

Escribe las soluciones y las pruebas el fichero `practica-08.scala`. Comprobaremos la compilación y ejecución correcta desde línea de comandos:

```
$ scala practica-08.scala
```

A continuación puedes ver un ejemplo de un ejercicio en el que se pide implementar una función que devuelva la suma de dos enteros elevada al cuadrado:

```
//
// Domingo Gallardo López
// 27/07/2015
//
// Ejercicio 0: Suma de cuadrados
//

def cuadrado(x: Int): Int = x * x

def sumaCuadrados1(x: Int, y: Int): Int = cuadrado(x) + cuadrado(y)

def sumaCuadrados2(x: Int, y: Int): Int = {
    val c1 = cuadrado(x)
    val c2 = cuadrado(y)
    c1 + c2
}

// pruebas

assert(sumaCuadrados1(1,1) == 2)
assert(sumaCuadrados2(1,1) == 2)
```

Algunas consideraciones importantes:

- Permitimos utilizar valores inmutables en las soluciones, utilizando la palabra reservada `val`.
- La forma de realizar pruebas en Scala será con la instrucción `assert` que lanza una excepción si recibe un valor falso.
- En los ejercicios puedes definir las funciones auxiliares que necesites, siempre que no se indique lo contrario.
- Recuerda los criterios de evaluación que anunciamos en el foro (enlace).

Ejercicio 1

Implementa en Scala la función recursiva `ordenada(lista: List[Int]): Boolean` que recibe como argumento una lista de valores de tipo `Int` y devuelve `true` si los números de la lista están ordenados de forma creciente y `false` en caso contrario. Suponemos listas de 1 o más elementos.

Ejercicio 2

Escribe las funciones

- `englobados(int1: (Int, Int), int2: (Int, Int)): Boolean`
- `intersectan(int1:(Int, Int), int2: (Int, Int)): (Int, Int)`

que comprueban si dos intervalos están englobados y devuelven el intervalo de intersección de uno con el otro (ver ejemplos en la práctica 1 de la asignatura). Los intervalos se representan como tuplas de dos `Int`. En el caso en que los dos intervalos no intersecten se debe devolver `null`.

Ejercicio 3

Escribe una versión recursiva pura y otra con recursión por la cola de la función `cuadradoLista(lista: List[Int]): List[Int]` que devuelve una lista con los números de la lista original elevados al cuadrado.

Ejercicio 4

Implementa la función `contienePatron(frase: String, patron: String): List[String]` que recibe una frase (cadena con palabras separadas por espacios) y un patrón a buscar en la frase. El patrón es sencillamente una cadena. Tiene que devolver una lista de palabras que contienen el patrón dado.

Consulta el API de Scala de operaciones sobre cadenas.

Ejemplo:

```
assert(contienePatron("El perro de roque no es un robot", "ro") ==
       List("perro", "roque", "robot"))
```

Ejercicio 5

Implementa la función `cuentaOcurrencias(lista1: List[String], lista2: List[String]): List[(String, Int, Int)]` que recibe dos listas de cadenas y devuelve una lista de tuplas de tipo `(String, Int, Int)`. Las tuplas representan el número de veces que aparece la cadena en la primera y la segunda lista.

Ejemplo:

```
assert(cuentaOcurrencias(List("pera", "melón", "pera", "manzana", "orégano"),
                         List("perejil", "pera", "hierbabuena", "orégano", "perejil")) ==
       List(("pera", 2, 1),
            ("melón", 1, 0),
            ("manzana", 1, 0),
            ("orégano", 1, 1),
            ("perejil", 0, 2),
            ("hierbabuena", 0, 1))
```

Construye funciones auxiliares y utiliza los métodos del API de la clase List.

Antonio Botía, Domingo Gallardo, Cristina Pomares