Ejercicios Prácticos de Programación Declarativa

Sesión de laboratorio 2

Curso 2019/20

- Realizad los siguientes ejercicios en un mismo fichero .hs.
- Escribid vuestro nombre al comienzo del fichero como líneas comentadas.
- Incluid comentarios significativos.
- Subid el fichero al Campus Virtual antes de que acabe la clase. Es suficiente con que lo suba uno si lo hacéis entre dos.

1. Definid funciones recursivas en Haskell para calcular las siguientes expresiones:

   a) La lista de los cuadrados de los números naturales entre 0 y n (o sea, $[0, 1, 4, 9, \ldots, n^2]$).

   b) La lista anterior, pero con cada número emparejado con su cuadrado y en orden inverso ($[(n, n^2), \ldots, (2, 4), (1, 1), (0, 0)]$).

   c) La suma $\sum_{i=1}^{i=100} i \cdot \mid sen(i) \mid$.

   d) El número de potencias de 3 que sean menores que n y acaben en 67.

   e) La suma de los números menores que 1000 que sean múltiplos de 3 o 5.

2. Programa, utilizando funciones de orden superior predefinidas, las siguientes funciones de orden superior. No olvides declarar sus tipos:

   - `filter2 xs p q` = (us, vs) donde us son los elementos de xs que cumplen p y vs los que cumplen q.
   - `filters xs ps` = $[xs_1, \ldots, xs_n]$, donde xs_i son los elementos de xs que cumplen p_i, supuesto que ps es $[p_1, \ldots, p_n]$.
   - `mapx x [f0,f1,...,fn] = [f0 x,f1 x,...,fn x]`.
   - `iguales f g n m` $\Leftrightarrow f\ x = g\ x$, para todo $n \leq x \leq m$.
   - `cuantos p xs` = número de elementos de la lista xs que cumplen la propiedad p.
   - `menorA n m p` = menor x con $n \leq x \leq m$ que verifica p.
   - `mayor n p` = mayor $x \leq n$ que verifica p.
   - `ex n m p` $\Leftrightarrow$ existe x con $n \leq x \leq m$ que verifica p.

3. Define mediante `foldr` o `foldl`, en lugar de mediante recursión explícita, las siguientes funciones: `last`, `reverse`, `all`, `minimum`, `map`, `filter`, `takeWhile`, `(++)`. Expresa mediante λ-expresiones el primer argumento de la la función `fold` que utilices.

4. Programa, indicando los tipos, las siguientes variantes de `foldl` y `foldr`, que operan con listas no vacías y no usan valor acumulado inicial:

   - $foldr1\ \bigoplus\ [x_1, \ldots, x_n] = x_1 \bigoplus x_2 \bigoplus \ldots \bigoplus x_n$ (con $\bigoplus$ asociando por la derecha)
   - $foldl1\ \bigoplus\ [x_1, \ldots, x_n] = x_1 \bigoplus x_2 \bigoplus \ldots \bigoplus x_n$ (con $\bigoplus$ asociando por la izquierda)

5. Programa al menos tres de los apartados del primer ejercicio utilizando funciones de orden superior en lugar de recursión explícita.