Ejercicios Prácticos de Programación Declarativa

Sesión de laboratorio 2

Curso 2022/23

- Realiza los siguientes ejercicios en un mismo fichero .hs.
- Incluye comentarios significativos sobre tus funciones y para contestar a las preguntas formuladas.
- **Declara los tipos de las funciones que definas**.

1. Programa en Haskell las siguientes funciones cuyo argumento es un entero no negativo n, sin usar recursión explícita sino **funciones de orden superior** predefinidas en Haskell.

   a) La lista los n primeros números naturales pares, ordenada de menor a mayor, usando **map**.

   b) La lista de pares formados por un número natural como primera componente del par y su cuadrado como segunda, ordenados desde el número natural n hasta 0: $([(n, n^2), \ldots, (2, 4), (1, 1), (0, 0)])$.

   c) La lista con las n primeras potencias de 3, usando `iterate`.

   d) La suma de los números menores que n que sean múltiplos de 3 o 5, usando **fold**.

   e) El primer número primo mayor que n. Intenta hacerlo de varias formas.

2. Programa las siguientes funciones de orden superior, utilizando funciones de orden superior predefinidas en Haskell. Los tipos de las variables n y m usadas en estas funciones tienen que estar en la clase `Enum`.

   a) `iguales f g n m` = `True` si y solo si `f` x = `g` x, para todo $n \leq x \leq m$.

   b) `menorA n m p` = menor x con $n \leq x \leq m$ que verifica `p`. ¿Qué ocurre si no existe tal x?

   c) `menor n p` = menor x con $x \geq n$ que verifica `p`. ¿Qué ocurre si no existe tal x? Utiliza ahora esta función para hallar el primer número primo mayor que n.

   d) `mayorA n m p` = mayor x con $n \leq x \leq m$ que verifica `p`. ¿Qué ocurre si no existe tal x?

   e) `pt n m p` = `True` si y solo si todos los x con $n \leq x \leq m$ verifican `p`.

3. Programa las siguientes funciones de orden superior, utilizando funciones de orden superior predefinidas en Haskell:

   a) `filter2 xs f g = [us,vs]` donde `us` es la lista resultante de aplicar `f` a los elementos de la lista `xs` y `vs` la lista resultante de aplicar `g` a los elementos de `xs`.

   b) `partition p xs = (us,vs)`, donde `us` son los elementos de la lista `xs` que cumplen `p` y `vs` son el resto.

   c) `mapx x [f0,f1,...,fn] = [f0 x,f1 x,...,fn x]`.

   d) `filter1 xss p = [ys1,...,ysn]`, donde si `xss` es la lista de listas `[xs1,...,xsn]`, entonces para todo $i, 1 \leq i \leq n$, `ysi` es la lista con los elementos de la lista `xsi` que cumplen la propiedad `p`.

   e) `filters xs ps = [ys1,...,ysn]`, donde para todo $i, 1 \leq i \leq n$, `ysi` es la lista con los elementos de la lista `xs` que cumplen la propiedad `pi`, supuesto que `ps=[p1,...,pn]`.