Programación Declarativa

Sesión de laboratorio 3

Curso 2022/23

- Realiza los siguientes ejercicios en un mismo fichero .hs.
- Escribe tu nombre al comienzo del fichero como líneas comentadas. Incluye comentarios significativos y no olvides `declarar los tipos` de las expresiones que definas.
- Sube el fichero al Campus Virtual antes de que acabe la clase.

1. Define mediante `foldr` o `foldl`, en lugar de mediante recursión explícita, las siguientes funciones predefinidas en Prelude. Expresa mediante λ-expresiones el primer argumento de la función `fold` que utilices.
   - `last`,
   - `reverse`,
   - `any`,
   - `minimum`,
   - `map`,
   - `takeWhile`.
2. Define una expresión Haskell para representar la lista $[1, -2, 3, -4, \ldots 99, -100]$ usando `foldl`.
3. Define una expresión Haskell, usando solo las funciones del preludio y/o λ-expresiones si te son de utilidad, pero no otras funciones auxiliares ni listas intensionales, para representar la lista $[\ [\ ], [1], [2, 2], [3, 3, 3], [4, 4, 4, 4], \ldots]$.
4. Expresa mediante una lista intensional la lista infinita:
   $[(0, 0), (0, 1), (1, 0), (0, 2), (1, 1), (2, 0), (0, 3), (1, 2), \ldots]$, que representa una enumeración (por diagonalización) de todas los pares de números naturales.
5. Programa la siguiente función, usando orden superior o listas intensionales.

   `permuta xs` = lista de todas las permutaciones de la lista `xs`. Por ejemplo:
   `permuta [1,2,3]=[[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]`.
6. Programa la siguiente función sin utilizar recursión explícita, sino alguna función de la familia `fold`.

   `cuestasPos xs` = `ys`, donde si `xs` es una lista no vacía de elementos que admiten un orden, `ys` es la lista de pares `(x,ps)`, tales que `x` es un elemento de `xs` *en cuesta en la lista* y `ps` es la posición que ocupa `x` en `xs` . Un elemento en cuesta en una lista es un elemento que es estrictamente mayor que el anterior a él en la lista. Por ejemplo:
   `cuestasPos [-3,-2,1,0,-1,1] = [(-2,1),(1,2),(1,5)]`.