Programación Declarativa

Sesión de laboratorio 4

Curso 2021/22

1. Define un tipo `Punto` para representar coordenadas enteras en una cuadrícula con filas y columnas numeradas de 0 a 100. La primera componente de cada coordenada representa la fila y la segunda la columna. Define un tipo enumerado `Direccion` con cuatro valores que representen movimientos (*arriba, abajo, izquierda, derecha*). Este tipo debe ser instancia de `Eq, Ord, Show` heredando los métodos de cada clase. Define las siguientes funciones:

   a) `mueve p m` devuelve el punto al que se mueve el punto `p` al aplicarle el movimiento `m`.

   b) `destino p ms` devuelve el punto final al que se llega desde el punto `p`, al aplicar sucesivamente los movimientos de la lista de movimientos finita `ms`. Utiliza `mueve` y un `fold`.

   c) `trayectoria p ms` devuelve la lista de puntos por los que se pasa al aplicar la lista de movimientos finita `ms` al punto del plano `p`. Utiliza también un `fold`.

2. Define un tipo `Nat` para representar números naturales con la aritmética de Peano. Es decir, toda expresión de tipo `Nat` será `Cero` o el sucesor de un elemento de `Nat` (expresión de la forma `Suc e` con `e::Nat`. Declara el tipo como instancia de `Eq` y `Ord` usando `deriving`.

   a) Define operadores infijos para calcular la suma y el producto de elementos de `Nat`.

   b) Define una función `natToInt` que convierta una expresión de tipo `Nat` en su correspondiente valor numérico en el tipo `Int`.

   c) Utiliza `natToInt` para declarar `Nat` como instancia de `Show`. Solo es necesario que definas `show` de manera que para cada expresión de tipo `Nat` muestre el número correspondiente.

3. Define un tipo para representar números complejos y declárarlo como instancia de la clase `Eq` usando `deriving`.
   Declárarlo como instancia de *Show* de modo que, por ejemplo, `show` de la expresión Haskell que represente al complejo $2 + 3i$ sea "`2+3i`"(análogamente "`2-3i`"para $2 - 3i$).
   Redefine los métodos `(+)`, `(-)` y `(*)` para definir el tipo de los complejos como instancia de `Num`. Estas funciones deberán expresar las operaciones aritméticas entre números complejos.
   Redefine el método `(/)` para definir este tipo como instancia de `Fractional`.

4. Define una clase de tipos `Medible` que disponga de un método `medida::a -> Int` que se pueda aplicar a cada tipo `a` de dicha clase. Declara algunos tipos como instancia de la clase `Medible`, por ejemplo `Bool`, `[a]`, `(a,b)`, definiendo la función `medida` para cada uno de ellos, como te parezca más oportuno.
   Por ejemplo, `medida [e1,...,en] = medida e1 +...+ medida en`.