Programación Declarativa

Sesión de laboratorio 4

Curso 2020/21

- Realiza los siguientes ejercicios individualmente 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 un tipo enumerado `Direccion` con cuatro valores que representen movimientos (*arriba, abajo, izquierda, derecha*) por una cuadrícula en el plano con coordenadas enteras. Convertirlo en instancia de `Eq, Ord, Show` usando `deriving`. Definie una función `destino` que al aplicarse a un punto del plano y una lista de movimientos, devuelva el punto final al que se llega.

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`.

   - Define operadores infijos para calcular la suma y el producto de elementos de `Nat`.
   - Define una función `natToInt` que convierta una expresión de tipo `Nat` en su equivalente en el tipo `Int`.
   - Utiliza `natToInt` para declarar `Nat` como instancia de `Show`. *(Para declarar un tipo* `T` *como instancia de Show basta con definir la función* `show::T->String`*, no hace falta definir las otras funciones de la clase Show).*

3. Define un tipo para representar números complejos y declararlo como instancia de las clases `Eq`, `Num`, y `Show` usando `deriving` solo cuando sea conveniente. Por ejemplo, `show` del término Haskell que represente al complejo $2 + 3i$ será el string "`2+3i`"(análogamente "`2-3i`"para $2 - 3i$). Tendrás que redefinir los métodos de `Num` para expresar las operaciones aritméticas entre números complejos, redefine al menos `(+)`, `(-)` y `(*)`.

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`, aunque puedes elegir otra definición.