Práctico 2 de Programación Funcional.

UdelaR/FCien/CMat

xx al yy de septiembre de 2016.

1. Dados tipos de datos `a`, `b`, `c` y `f::c -> a` y `g::c -> b` funciones calculables, probar que existe una única `(f, g) :: c -> (a, b)` función calculable tal que `f = fst.(f, g)` y `g = snd.(f, g)` (*propiedad universal del producto*). Definirla en Haskell y verificar que la definición es correcta para $\perp$. Definir en Haskell una función (polimorfa) `pair` que calcule `(f, g)` a partir de `f` y `g`. ¿Cuál es el tipo de `pair`?

2. Dados tipos de datos `a`, `b`, `c`, `d` y `f::a -> c` y `g::b -> d` funciones calculables, probar que existe una única función calculable `f X g` tal que `fst.(f X g) = f.fst` y `snd.(f X g) = g.snd`. Definirla en Haskell y verificar que la definición es correcta para datos parciales i.e.: $\perp$, `(`$\perp$`, y)` y `(x, `$\perp$`)` con `x::a` e `y::b`. Definir en Haskell una función (polimorfa) `cross` que calcule `f X g` a partir de `f` y `g`. ¿Cuál es el tipo de `cross`?. Demostrar que `(cross f g).(cross h k) = cross (f.h) (g.k)`.

3. Definir un tipo de datos `Triple a b c` que corresponde a ternas y funciones `fst`, `snd`, `thrd` que devuelvan respectivamente la primera, segunda y tercera coordenada de una terna. Enunciar y demostrar la propiedad universal para el tipo `Triple a b c`.

4. ¿Es posible declarar `(a, b)` como miembro de la clase `Enum` sabiendo que `a` y `b` son miembros de `Enum`? Hacer las definiciones correspondientes en Haskell.

5. Usando las funciones de los ejercicios 1 y 3, definir el tipo `Triangle` como ternas de pares del tipo `(Float, Float)` que corresponden a las coordenadas de los vértices de un triángulo en el plano euclídeo. Definir sobre ese tipo las funciones `area`, `permeter` y `barycentre` que calculan respectivamente el área, el perímetro y las coordenadas del baricentro de un triángulo. ¿Cuál es el tipo de cada una de estas funciones? ¿Cómo calculan con datos parciales?

6. Definir el tipo `data Angle = MkAngle Float`. Declarar `Angle` como instancia de la clase `Eq`. Definir la función `normalize :: Angle -> Angle` que, dado un ángulo, lo exprese como un ángulo entre 0 y 2π. Definir la suma y el producto de ángulos, de modo que el resultado esté normalizado. ¿Tiene sentido derivar automáticamente `Angle` como instancia de

la clase `Ord`? Justificar. ¿Cómo declarar `Angle` como instancia de la clase `Ord` para que el orden capture la noción geométrica de subángulo, esto es: $\alpha < \beta$ si y sólo si α es congruente con una parte propia de β?

7. Definir el tipo `data List a = Nil | Cons a (List a)`. Derivar automáticamente a `List a` como instancia de `Eq, Ord`. Esta derivación declara, a partir de tipos `a, b` de la clase `Ord`, a `D a b` como instancia de la clase `Ord`. Describir matemáticamente el orden definido por la declaración. Definir el tipo `data D a b = C1 | C2 a | C3 a b | C4 a (D a b) b` y hacer lo mismo que para `List a`, describiendo el orden derivado automáticamente. **Sug.:** en ambos ejemplos, investigar primero qué sucede con datos que usan, en su nivel exterior, distintos constructores. Luego observar qué sucede si en el nivel exterior usan el mismo constructor.

8. Consideramos la implementación `Set4` de los conjuntos no vacíos de a lo sumo 4 enteros como las 4-uplas de `(Int, Int, Int, Int)`. Definir el tipo `Set4`. Declarar a `Set4` como instancia de las clases `Eq, Ord`, de modo que la igualdad y el orden sean respectivamente la igualdad y la inclusión de conjuntos. **Sug.:** definir una función auxiliar que ordene una cuaterna en sentido creciente. ¿Qué sucede si usamos la derivación automática?

   Hacer este ejercicio declarando `Set4` mediante `data` y mediante `newtype`. ¿Cuál implementación es más eficiente? ¿Es posible usar una declaración `type`? Justificar.