Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Escuela de Ciencias Exactas y Naturales
Departamento de Ciencia de la Computación
Análisis de Lenguajes de Programación

Práctica 5

Functores

1. Demostrar que los siguientes tipos de datos son functores. Es decir, dar su instancia de la clase Functor correspondiente y probar que se cumplen las leyes de los functores.

a) **data** Pair a = P (a, a)

b) **data** Tree a = Empty | Branch a (Tree a) (Tree a)

c) **data** GenTree a = Gen a [GenTree a]

d) **data** Cont a = C $((a \to \text{Int}) \to \text{Int})$

2. Probar que las siguientes instancias no son correctas (no cumplen las leyes de los functores).

a) **data** Func a = Func $(a \to a)$

```
instance Functor Func where
  fmap g (Func h) = Func id
```

b) **data** Br b a = B b (a, a)

```
instance Functor (Br b) where
  fmap f (B x (y, z)) = B x (f z, f y)
```

Functores aplicativos

3. Dar la instancia de Applicative para:

a) Either e, con e fijo.

b) $(\to)$ r, con r fijo.

4. Las funciones liftAx aplican una función a x argumentos contenidos en una estructura. Usando los operadores de la clase Applicative, dar las definiciones de:

a) liftA2 :: Applicative $f \Rightarrow (a \to b \to c) \to f\ a \to f\ b \to f\ c$
Por ejemplo, liftA2 $(,)$ (Just 3) (Just 5) evalúa a Just $(3, 5)$.

b) liftA5 :: Applicative $f \Rightarrow (a \to b \to c \to d \to e \to k) \to f\ a \to f\ b \to f\ c \to f\ d \to f\ e \to f\ k$

5. Definir una función sequenceA :: Applicative $f \Rightarrow [f\ a] \to f\ [a]$, que dada una lista de acciones de tipo $f\ a$, siendo f un funtor aplicativo, transforme la lista una acción de tipo $f\ [a]$.

Uso de Mónadas y notación **do**

6. Probar que toda mónada es un functor, es decir, proveer una instancia

```
instance Monad m ⇒ Functor m where
  fmap ...
```

y probar que las leyes de los functores se cumplen para su definición de fmap.

7. Dado el siguiente tipo de datos para representar expresiones matemáticas:

```
data Expr a = Var a | Num Int | Add (Expr a) (Expr a)
```

´

a) Dar la instancia de Monad para Expr y probar que es una mónada.

b) Dar el tipo y la definición de la función g de manera que Add (Var "y") (Var "x") $\gg\!=$ g evalúe a la expresión Add (Mul (Var 1) (Num 2)) (Var 1).

c) Explicar qué representa el operador $\gg\!=$ para este tipo de datos.

8. Definir las siguientes funciones:

a) mapM :: Monad $m \Rightarrow (a \to m\ b) \to [a] \to m\ [b]$, tal que mapM f xs aplique la función monádica f a cada elemento de la lista xs, retornando la lista de resultados encapsulada en la mónada.

b) foldM :: Monad $m \Rightarrow (a \to b \to m\ a) \to a \to [b] \to m\ a$, análogamente a foldl para listas, pero con su resultado encapsulado en la mónada. *Ejemplo:*

```
foldM f e1 [x1, x2, x3] = do e2 ← f e1 x1
                             e3 ← f e2 x2
                             f e3 x3
```

9. Escribir el siguiente fragmento de programa monádico usando notación **do**.

$$(m \gg\!= \lambda x \to h\ x) \gg\!= \lambda y \to f\ y \gg\!= \lambda z \to \mathsf{return}\ (g\ z)$$

10. Escribir el siguiente fragmento de programa en términos de $\gg\!=$ y return.

```
do x ← (do z ← y
           w ← f z
           return (g w z))
   y ← h x 3
   if y then return 7
        else do z ← h x 2
                return (k z)
```

11. Escribir las leyes de las mónadas usando la notación **do**.

I/O Monádico

12. Escribir y **compilar** un programa (usando `ghc` en lugar de `ghci`) que imprima en pantalla la cadena `"Hola mundo!"`.

13. Dar una definición de la función getChars :: Int → IO String, que dado n lea n caracteres del teclado, usando las funciones sequenceA y replicate.

14. Escribir un programa interactivo que implemente un juego en el que hay que adivinar un número secreto predefinido. El jugador ingresa por teclado un número y la computadora le dice si el número ingresado es menor o mayor que el número secreto o si el jugador adivinó, en cuyo caso el juego termina. Ayuda: para convertir una String en Int puede usar la función read :: String → Int.

15. El juego nim consiste en un tablero de 5 filas numeradas de asteriscos. El tablero inicial es el siguiente:

1 : * * * * *
2 : * * * *
3 : * * *
4 : * *
5 : *

Dos jugadores se turnan para sacar una o mas estrellas de alguna fila. El ganador es el jugador que saca la última estrella. Implementar el juego en Haskell. Ayuda: para convertir una String en Int puede usar la función read :: String → Int.

16. Un programa pasa todos los caracteres de un archivo de entrada a mayúsculas y los guarda en un archivo de salida. Hacer un programa compilado que lo implemente tomando dos argumentos en la línea de comandos, el nombre de un archivo de entrada y el nombre de un archivo de salida.