Programación Funcional

Segunda Prueba Escrita - 2024

Nombre:

CI:

Número:

---

1. Dada la siguiente definición:

   $$foo\ f\ x\ y\ z\ w = foldr\ (:)\ [\,]\ \$\ zipWith\ f\ [x == z, y == w]\ [x, z]$$

   El tipo más general es:

   (a) $foo :: (Eq\ a) \Rightarrow (Bool \to a \to c) \to a \to a \to a \to a \to [c]$
   (b) $foo :: (Eq\ a, Eq\ b) \Rightarrow (Bool \to a \to c) \to a \to b \to a \to b \to [c]$
   (c) $foo :: (Eq\ a, Eq\ b) \Rightarrow (Bool \to a \to c) \to a \to b \to a \to b \to c$
   (d) $foo :: (Eq\ a) \Rightarrow (Bool \to a \to c) \to a \to a \to a \to a \to c$

   Respuesta: b)

2. Considere la siguiente definición de *dropL xs ys*.

   $$dropL\ [\,]\ \quad ys\ \quad = ys$$
   $$dropL\ (x : xs)\ (y : ys) = dropL\ xs\ ys$$

   Si se asume que *xs* es una lista finita y tiene igual o menos elementos que *ys*, indique cuál de las siguientes definiciones **NO** es equivalente a *dropL*.

   (a) $dropL\ xs\ ys = foldr\ (const\ id)\ ys\ xs$
   (b) $dropL\ xs\ ys = drop\ (length\ xs)\ ys$
   (c) $dropL\ xs\ ys = foldl\ (flip \circ const\ \$\ tail)\ ys\ xs$
   (d) $dropL\ xs\ ys = foldr\ (const\ tail)\ ys\ xs$

   Respuesta: a)

3. Dada la siguiente definición:

   $$automap\ f = concat \circ map\ f \circ f$$

   ¿Cuál de las siguientes opciones **NO** es correcta?

   (a) *automap head* está mal tipada
   (b) *automap* (:[ ]) 3 retorna [3]
   (c) *automap* (*iterate id*) está mal tipada
   (d) El tipo más general de *automap* es $(a \to [a]) \to a \to [a]$

   Respuesta: c)

4. Dada la siguiente definición:

$$pflip\ f = uncurry\ (flip\ (curry\ f))$$

¿Cuál de las siguientes opciones es correcta?

(a) El tipo de *pflip snd* es $(a, c) \to c$

(b) *pflip fst* $(3, 4)$ retorna 3

(c) *pflip id* $(3, 4)$ retorna $(4, 3)$

(d) El tipo más general de *pflip* es $((a, b) \to c) \to (a, b) \to c$

Respuesta: c)

5. Considere la siguiente definición alternativa de *take*, donde *xs* puede ser tanto finita como infinita.

$$\begin{array}{l} take\ n\ xs = ftake\ [box\ m\ x \mid (x, m) \leftarrow zip\ xs\ [1\,..]] \\ \quad \textbf{where}\ box\ m\ x \mid m \leqslant n \quad = [x] \\ \qquad\qquad\qquad\quad \mid otherwise = [\,] \end{array}$$

¿Cuál de las siguientes implementaciones de $ftake :: [[a]] \to a$ es correcta?

(a) $ftake = concat \circ filter\ (not \circ null)$

(b) $ftake = concat$

(c) $ftake = map\ head \circ filter\ (not \circ null)$

(d) $ftake = map\ head \circ takeWhile\ (not \circ null)$

Respuesta: d)

6. Dadas las siguientes definiciones:

$$\textbf{data}\ T\ a = L\ a \mid N\ (T\ a)\ a\ (T\ a)$$

$$\begin{array}{l} f\ (L\ a) \quad\ xs = xs \\ f\ (N\ l\ a\ r)\ xs = f\ l\ (a : f\ r\ xs) \end{array}$$

$$\begin{array}{l} h\ (L\ a) \quad\ xs = a : xs \\ h\ (N\ l\ a\ r)\ xs = h\ r\ (h\ l\ xs) \end{array}$$

$$t1 = N\ t2\ 1\ (N\ (L\ 2)\ 3\ (L\ 4))$$

$$t2 = N\ (N\ t1\ 5\ t1)\ 6\ (L\ 7)$$

¿Cuál de las siguientes opciones **NO** es correcta?

(a) *take* $4\ \$\ f\ t2\ (h\ t2\ [\,])$ diverge

(b) *take* $4\ \$\ h\ t2\ (f\ t2\ [\,])$ diverge

(c) *take* $4\ \$\ f\ t1\ (h\ t1\ [\,])$ diverge

(d) *take* $4\ \$\ h\ t1\ (f\ t1\ [\,])$ retorna $[4, 2, 7, 4]$

Respuesta: b)

7. Dadas la siguientes definiciones:

$$loop = pzip\ [0..]\ (map\ snd\ loop)$$

$$\begin{aligned} pzip\ [\,]\ \_ &= [\,] \\ pzip\ (x : xs)\ ys &= (x, head\ ys) : pzip\ xs\ (tail\ ys) \end{aligned}$$

Para cada una de las siguientes expresiones indique el resultado de su evaluación o si la misma diverge (si pone diverge en todas las opciones anula la pregunta).

(a) $length\ \$\ take\ 3\ \$\ map\ snd\ loop$ — 3

(b) $sum\ \$\ take\ 3\ \$\ map\ fst\ loop$ — 3

(c) $sum\ \$\ take\ 3\ \$\ map\ snd\ loop$ — diverge

(d) $snd\ \$\ head\ \$\ dropWhile\ (\lambda(x, y) \to x < 4)\ loop$ — diverge

(e) $fst\ \$\ head\ \$\ dropWhile\ (\lambda(x, y) \to x < 4)\ loop$ — 4

(f) $map\ fst\ \$\ take\ 3\ loop$ — [0,1,2]

(g) $filter\ (\lambda(x, y) \to x < 0)\ loop$ — diverge

(h) $foldl\ (\lambda a\ (x, y) \to x + a)\ 0\ loop$ — diverge

8. Dadas las siguientes definiciones:

$$\textbf{data}\ BTree\ a = Fork\ (BTree\ a)\ (BTree\ a) \mid Leaf\ a\ \ \textbf{deriving}\ Show$$

$$\textbf{data}\ Dir = Der \mid Izq$$

$$\begin{aligned} go\ Izq\ \ (Just\ (Fork\ l\ \_)) &= Just\ l \\ go\ Der\ (Just\ (Fork\ \_\ r)) &= Just\ r \\ go\ \_\ \ \_ &= Nothing \end{aligned}$$

$$goPath = foldr\ (\circ)\ Just \circ map\ go \circ reverse$$

$$\begin{aligned} t &= (Fork\ (Fork\ (Fork\ (Leaf\ 1)\ (Leaf\ 2))\ (Leaf\ 3))\ (Leaf\ 4)) \\ tinf &= Fork\ tinf\ (Leaf\ 9) \end{aligned}$$

¿Cuál de las siguientes opciones **NO** es correcta?

(a) $(goPath\ (repeat\ Der)\ t)$ diverge

(b) $(goPath\ [Der]\ tinf)$ diverge

(c) $(goPath\ [Izq, Izq, Der]\ t)$ retorna $(Just\ (Leaf\ 2))$

(d) $(goPath\ [Der, Der, Der]\ t)$ retorna $(Nothing)$

Respuesta: b)

9. Implemente, sin usar recursión, el operador

$$(\texttt{|\$>}) :: (a \to Bool) \to (a \to b) \to ([a] \to [b])$$

tal que p `|$>` f retorna una función que dada una lista xs, aplica f solo a aquellos elementos de xs que cumplen el predicado p.
Por ejemplo, $((>1)$ `|$>` $(+1))\ [5,1,2,0,4]$ retorna $[6,3,5]$.

$$p \texttt{ |\$> } f = map\ f \circ filter\ p$$

10. Utilizando *recursión explícita*, escriba una función *tail recursiva*:

$$appList :: a \to [a \to a] \to a$$

que dado un elemento z y una lista de funciones $[f_1, f_2, ..., f_n]$, aplique las funciones al elemento, en el orden dado. Esto es, $appList\ z\ [f_1, f_2, ..., f_n]$ retorna $f_n\ (...f_2\ (f_1\ z)...)$. En el caso de la lista vacía retorna directamente el elemento.
Por ejemplo, $appList\ 2\ [(+1), (*2), (+8)] = 14$.

$$\begin{aligned} appList\ z\ [\,] &= z \\ appList\ z\ (f : fs) &= appList\ (f\ z)\ fs \end{aligned}$$

Implemente la misma función, pero *como un foldl*.

$$appList = foldl\ (flip\ (\$))$$