

Programación Funcional

Segunda Prueba Escrita - 2024

Nombre:

CI:

Número:

1. Considere la siguiente definición de $dropL\ xs\ ys$.

$$\begin{aligned} dropL\ []\ ys &= ys \\ dropL\ (x : xs)\ (y : ys) &= dropL\ xs\ ys \end{aligned}$$

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

Respuesta: d)

2. Dada la siguiente definición:

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

El tipo más general es:

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

Respuesta: c)

3. Dada la siguiente definición:

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

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

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

Respuesta: d)

4. Dadas las siguientes definiciones:

data $T\ a = L\ a \mid N\ (T\ a)\ a\ (T\ a)$

$f\ (L\ a)\ xs = xs$
 $f\ (N\ l\ a\ r)\ xs = f\ l\ (a : f\ r\ xs)$

$h\ (L\ a)\ xs = a : xs$
 $h\ (N\ l\ a\ r)\ xs = h\ r\ (h\ l\ xs)$

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

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

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

- (a) $take\ 4\ \$\ h\ t1\ (f\ t1\ [])$ retorna $[4, 2, 7, 4]$
- (b) $take\ 4\ \$\ f\ t2\ (h\ t2\ [])$ diverge
- (c) $take\ 4\ \$\ h\ t2\ (f\ t2\ [])$ diverge
- (d) $take\ 4\ \$\ f\ t1\ (h\ t1\ [])$ diverge

Respuesta: c)

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

$take\ n\ xs = ftake\ [box\ m\ x \mid (x, m) \leftarrow zip\ xs\ [1..]]$
where $box\ m\ x \mid m \leq n = [x]$
 $\mid otherwise = []$

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

- (a) $ftake = map\ head \circ takeWhile\ (not \circ null)$
- (b) $ftake = concat$
- (c) $ftake = map\ head \circ filter\ (not \circ null)$
- (d) $ftake = concat \circ filter\ (not \circ null)$

Respuesta: a)

6. Dada la siguiente definición:

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

¿Cuál de las siguientes opciones es correcta?

- (a) $pflip\ id\ (3, 4)$ retorna $(4, 3)$
- (b) El tipo más general de $pflip$ es $((a, b) \rightarrow c) \rightarrow (a, b) \rightarrow c$
- (c) El tipo de $pflip\ snd$ es $(a, c) \rightarrow c$
- (d) $pflip\ fst\ (3, 4)$ retorna 3

Respuesta: a)

7. Dadas la siguientes definiciones:

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

$$pzip [] _ = []$$

$$pzip (x : xs) ys = (x, head ys) : pzip xs (tail ys)$$

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) $sum \$ take 3 \$ map fst loop$

3

(b) $length \$ take 3 \$ map snd loop$

3

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

diverge

(d) $fst \$ head \$ dropWhile (\lambda(x, y) \rightarrow x < 4) loop$

4

(e) $snd \$ head \$ dropWhile (\lambda(x, y) \rightarrow x < 4) loop$

diverge

(f) $foldl (\lambda a (x, y) \rightarrow x + a) 0 loop$

diverge

(g) $filter (\lambda(x, y) \rightarrow x < 0) loop$

diverge

(h) $map fst \$ take 3 loop$

[0,1,2]

8. Dadas las siguientes definiciones:

data *BTree* *a* = *Fork* (*BTree* *a*) (*BTree* *a*) | *Leaf* *a*
deriving *Show*

data *Dir* = *Der* | *Izq*

$$go\ Izq\ (Just\ (Fork\ l\ _)) = Just\ l$$

$$go\ Der\ (Just\ (Fork\ _\ r)) = Just\ r$$

$$go\ _\ _ = Nothing$$

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

$$t = (Fork\ (Fork\ (Fork\ (Leaf\ 1)\ (Leaf\ 2))\ (Leaf\ 3))\ (Leaf\ 4))$$

$$tinf = Fork\ tinf\ (Leaf\ 9)$$

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

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

(b) $(goPath [Der, Der, Der] t)$ retorna $(Nothing)$

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

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

Respuesta: d)

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

$$appList :: a \rightarrow [a \rightarrow a] \rightarrow a$$

que dado un elemento z y una lista de funciones $[f_1, f_2, \dots, f_n]$, aplique las funciones al elemento, en el orden dado. Esto es, $appList\ z\ [f_1, f_2, \dots, f_n]$ 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*.

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

10. Implemente, sin usar recursión, el operador

$$(|\$\>) :: (a \rightarrow Bool) \rightarrow (a \rightarrow b) \rightarrow ([a] \rightarrow [b])$$

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 $[6, 3, 5]$.

$$p\ |\$\>\ f = map\ f \circ filter\ p$$