Programación Funcional

Prueba Escrita - 2022

Nombre:

CI:

Número de Prueba:

---

1. Dada la siguiente definición:

$$foo\ a\ b\ c = map\ (show \circ snd)\ \$\ zip\ [a, const\ c\ b, a + c]\ (repeat\ b)$$

   El tipo más general es:

   (a) $foo :: (Num\ a, Show\ a) \Rightarrow a \to a \to a \to [String]$
   (b) $foo :: (Num\ a, Show\ a) \Rightarrow a \to a \to a \to [a]$
   (c) $foo :: (Num\ a, Show\ b) \Rightarrow a \to b \to a \to [String]$
   (d) $foo :: (Num\ a, Show\ b) \Rightarrow a \to b \to a \to [b]$

   Respuesta: c)

2. Dada la siguiente definición:

$$twice\ f = f \circ f$$

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

   (a) El tipo más general de $(twice \circ twice)$ es $(a \to a) \to a \to a$
   (b) $(twice\ (twice\ (+1)))$ está mal tipada
   (c) $(twice\ fst)$ está mal tipada
   (d) El tipo más general de $(twice\ twice)$ es $(a \to a) \to a \to a$

   Respuesta: b)

3. Dadas las siguientes definiciones:

$$\begin{aligned} moo\ f\ k\ [\,] &= 0 \\ moo\ f\ k\ (x : xs) &\mid f\ k == x = 1 + moo\ f\ (f\ k)\ (tail\ xs) \\ &\mid otherwise = 0 \end{aligned}$$

   ¿Cuál de las siguientes afirmaciones es correcta?

   (a) El resultado de evaluar $moo\ id\ 1\ (take\ 10\ (repeat\ 1))$ es 10
   (b) Al intentar evaluar $moo\ (+2)\ 0\ [2\,..\,6]$ la ejecución da error
   (c) El resultado de evaluar $moo\ (+1)\ 1\ [1\,..\,6]$ es 1
   (d) El código no compila

   Respuesta: b)

4. Dada la siguiente definición:

$$intercalate :: [a] \to [[a]] \to [a]$$
$$intercalate\ xs = f \circ g \circ (h\ xs)$$

¿Cuál de las siguientes implementaciones de f, g y h hace que *intercalate* tome una lista (xs) y una lista de listas (xss), intercale xs entre los elementos de xss y concatene el resultado?

Ejemplos:
intercalate `","` [`"primero"`, `"segundo"`, `"tercero"`] = `"primero,segundo,tercero"`
intercalate $[1,2]$ $[[3,4],[5,6]] = [3,4,1,2,5,6]$
intercalate $[1,2]$ $[] = []$

(a) $f = concat$
$g = \lambda ys \to \textbf{if}\ null\ ys\ \textbf{then}\ ys\ \textbf{else}\ tail\ ys$
$h\ xs = foldr\ (\lambda x\ rs \to xs : x : rs)\ []$

(b) $f = id$
$g = concat$
$h\ xs = zipWith\ (+\!\!+)\ (repeat\ xs)$

(c) $f = \lambda ys \to \textbf{if}\ null\ ys\ \textbf{then}\ ys\ \textbf{else}\ tail\ ys$
$g = concat$
$h\ xs = map\ (xs+\!\!+)$

(d) $f = concat$
$g = id$
$h\ xs = map\ (xs+\!\!+)$

Respuesta: a)

5. Dada la siguiente definición:

$$\textbf{data}\ T\ a = D\ (T\ a)\ a\ (T\ a) \mid E$$
$$foo\ E \quad\quad \_\ a = a$$
$$foo\ (D\ l\ x\ r)\ f\ a = foo\ l\ f\ (foo\ r\ f\ (f\ x\ a))$$

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

(a) El tipo más general de *foo* es $T\ a \to (a \to b \to b) \to b \to b$

(b) Para todo t finito de tipo $T\ Int$, $foo\ t\ (+)\ 0$ retorna la suma de sus valores

(c) Para todo t finito de tipo $T\ a$, $foo\ t\ (:)\ []$ retorna una lista con sus valores ordenados de acuerdo a una recorrida izquierda-derecha-raiz

(d) Para todo t finito de tipo $T\ Int$, $foo\ t\ const\ 0$ retorna 0

Respuesta: d)

6. Dado el siguiente programa:

```
main = do x ← putStr "hola"
          y ← putStr "chau"
          putStr $ show (fst (x, y))
```

Al ejecutarlo imprime:

(a) `holachau()`

(b) `hola`

(c) `hola()`

(d) `()`

Respuesta: a)

7. Dada la siguiente definición:

$as = iterate\ (+1)\ 1$
$bs = as : zipWith\ drop\ as\ bs$

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) $take\ 5\ (map\ head\ bs)$ — [1,2,4,7,11]

(b) $take\ 5\ (bs\ !!\ 2)$ — [4,5,6,7,8]

(c) $head\ \$\ foldr\ (:)\ []\ (map\ sum\ bs)$ — diverge

(d) $head\ \$\ foldl\ (flip\ (:))\ []\ (map\ sum\ bs)$ — diverge

(e) $(\lambda xs \to take\ (head\ xs)\ xs)\ \$\ (head \circ tail)\ bs$ — [2,3]

(f) $fst \circ head\ \$\ zip\ as\ bs$ — 1

(g) $head \circ snd \circ head\ \$\ zip\ as\ bs$ — 1

(h) $take\ 2\ \$\ map\ (length \circ filter\ (<2))\ bs$ — diverge

8. Consideramos la siguiente definición de expresiones enteras con suma y multiplicación.

$$\begin{aligned}
&\textbf{data}\ Exp = Lit\ Int \mid Add\ Exp\ Exp \mid Mul\ Exp\ Exp\\
&foldE :: (Int \to a) \to (a \to a \to a) \to (a \to a \to a) \to Exp \to a\\
&foldE\ l\ _\ _\ (Lit\ x) \quad\ = l\ \ x\\
&foldE\ l\ a\ m\ (Add\ e1\ e2) = a\ \ (foldE\ l\ a\ m\ e1)\ (foldE\ l\ a\ m\ e2)\\
&foldE\ l\ a\ m\ (Mul\ e1\ e2) = m\ (foldE\ l\ a\ m\ e1)\ (foldE\ l\ a\ m\ e2)
\end{aligned}$$

Dada una expresión e finita ¿cuál de las siguientes afirmaciones **NO** es correcta?

(a) $foldE\ id\ (+)\ (*)\ e$ retorna el resultado de la evaluación de e

(b) $foldE\ (const\ 0)\ (\lambda x\ y \to x + y + 1)\ (+)\ e$ cuenta la cantidad de sumas que tiene e

(c) $foldE\ id\ const\ const\ e$ retorna el entero de más a la izquierda de e

(d) $foldE\ (const\ 0)\ Add\ Mul\ e$ cambia el valor de todos los literales enteros de e por el valor 0

Respuesta: d)

9. Implemente usando *recursión explícita* la función:

$$group :: Eq\ a \Rightarrow [a] \to [[a]]$$

que dada una lista de elementos comparables devuelve una lista de listas con los mismos elementos y en el mismo orden, pero agrupados en listas que contienen los elementos contiguos iguales. Ejemplos:

group `"Mississippi"` devuelve [”M”,”i”,”ss”,”i”,”ss”,”i”,”pp”,”i”]

group $[1,2,2,3,4,4,2,5,5,5,1]$ devuelve [[1],[2,2],[3],[4,4],[2],[5,5,5],[1]]

group $[\,]$ devuelve []

```
group []       = []
group (x : xs) = insGroup x (group xs)
  where insGroup x (ys@(y: _) : yss) | x == y    = (x : ys) : yss
                                     | otherwise = [x] : ys : yss
        insGroup x yss               = [x] : yss
```

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

```
group = foldr insGroup []
  where insGroup x (ys@(y: _) : yss) | x == y    = (x : ys) : yss
                                     | otherwise = [x] : ys : yss
        insGroup x yss               = [x] : yss
```

10. Implemente, sin usar recursión, la función *composeN*:

$$composeN :: (b \to b) \to Int \to b \to b$$

que dadas una función f y un entero n, compone f consigo mismo n veces. Por ejemplo, *composeN* $(+1)$ 4 1 devuelve 5.

```
composeN f n = foldr (∘) id $ take n (repeat f)
```