

Programación Funcional

Prueba Escrita - 2024

Nombre:

CI:

Número:

1. Dada la siguiente definición:

$$foo\ a\ b\ c\ d = foldr\ (+)\ c\ [fst\ a, \mathbf{if}\ b == snd\ a\ \mathbf{then}\ d\ \mathbf{else}\ d + 1]$$

El tipo más general es:

- (a) No tiene
- (b) $foo :: (Num\ a, Eq\ b, Num\ c) \Rightarrow (a, b) \rightarrow b \rightarrow c \rightarrow a \rightarrow c$
- (c) $foo :: (Num\ a, Eq\ a) \Rightarrow (a, a) \rightarrow a \rightarrow a \rightarrow a \rightarrow a$
- (d) $foo :: (Num\ a, Eq\ b) \Rightarrow (a, b) \rightarrow b \rightarrow a \rightarrow a \rightarrow a$

Respuesta: d)

2. Dada la siguiente definición:

$$rara = zipWith\ flip\ (repeat\ div)\ [0..]$$

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

- (a) $(rara\ !!\ 10)\ 10$, retorna 1
- (b) $(rara\ !!\ 0)\ 0$, da error de ejecución
- (c) $(rara\ !!\ 1)\ 0$, da error de ejecución
- (d) $(rara\ !!\ 0)\ 1$, da error de ejecución

Respuesta: c)

3. 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 $(a \rightarrow a) \rightarrow a \rightarrow a$
- (b) El tipo más general de $twice\ tail$ es $[a] \rightarrow [a]$
- (c) $twice\ twice$ está mal tipada
- (d) $twice\ head$ está mal tipada

Respuesta: c)

4. ¿Cuál de las siguientes definiciones no equivale a las otras tres:?

(a) $foo\ p = map\ (uncurry\ (+)) \circ filter\ (not \circ p \circ fst)$

(b) $foo\ p\ [] = []$
 $foo\ p\ ((x, y) : xs) \mid not\ (p\ x) = x + y : foo\ p\ xs$
 $\mid otherwise = foo\ p\ xs$

(c) $foo\ p\ xs = [x + y \mid (x, y) \leftarrow xs, not\ (p\ x)]$

(d) $foo\ p = zipWith\ (+) \circ filter\ (\lambda(x, -) \rightarrow not\ (p\ x))$

Respuesta: d)

5. Dada la siguiente definición:

```
data T a b = E | NA (T a b) a (T a b) | NB (T a b) b (T a b)
foo E           = ([], [])
foo (NA l x r) = let (ll, lr) = foo l
                  (rl, rr) = foo r
                  in (x : ll ++ rl, lr ++ rr)
foo (NB l x r) = let (ll, lr) = foo l
                  (rl, rr) = foo r
                  in (ll ++ rl, x : lr ++ rr)
```

Indique la opción correcta:

- (a) $foo\ (NA\ (NB\ (NA\ E\ 'a'\ E)\ True\ (NB\ E\ False\ E))\ 'g'\ (NA\ E\ 'w'\ E))$, retorna $(['w', 'a', 'g'], [False, True])$
- (b) $foo\ (NA\ (NB\ (NA\ E\ E\ E)\ E\ (NB\ E\ E\ E))\ E\ (NA\ E\ E\ E))$, no compila
- (c) $foo\ (NA\ (NB\ (NA\ E\ 1\ E)\ Nothing\ (NB\ E\ Nothing\ E))\ 2\ (NA\ E\ 3\ E))$, retorna $([1, 2, 3], [Nothing, Nothing])$
- (d) $foo\ (NA\ (NB\ (NA\ E\ 10\ E)\ 'a'\ (NB\ E\ 's'\ E))\ 9\ (NA\ E\ 89\ E))$, retorna $([9, 10, 89], ['a', 's'])$

Respuesta: d)

6. La función

$split :: [a] \rightarrow ([a], [a])$

divide una lista en dos listas colocando sus elementos de forma alternada. Por ejemplo, $split\ [2, 4, 6, 8, 7]$ retorna $([2, 6, 7], [4, 8])$.

Se puede implementar usando $foldl$ de la siguiente forma:

$split = foldl\ step\ ([], []) \circ ini$

Indique la opción que permite una implementación correcta de $split$.

(a) $ini = id$
 $step\ (ls, rs)\ x = (x : ls, rs)$

(b) $ini = id$
 $step\ (ls, rs)\ x = (x : rs, ls)$

(c) $ini = reverse$
 $step\ (ls, rs)\ x = (x : rs, ls)$

(d) $ini = reverse$
 $step\ (ls, rs)\ x = (x : ls, rs)$

Respuesta: c)

7. Dadas la siguientes definiciones:

$$\begin{aligned} \text{loop} &= \text{tail loop} \\ \text{ls} &= (\text{head loop}) : \text{rs} \\ \text{rs} &= 1 : \text{map } (+1) \text{ ls} \end{aligned}$$

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) head rs | 1 |
| (b) $(\text{head} \circ \text{tail} \circ \text{tail}) \text{ rs}$ | 2 |
| (c) $\text{length } (\text{take } 3 \text{ rs})$ | 3 |
| (d) $(\text{head} \circ \text{tail}) \text{ rs}$ | diverge |
| (e) $\text{take } 4 \$ \text{ map fst } \$ \text{ filter } ((/= 0) \circ ('mod' 2) \circ \text{fst}) \$ \text{ zip } [1..] \text{ rs}$ | [1,3,5,7] |
| (f) $\text{take } 4 \$ \text{ map snd } \$ \text{ filter } ((/= 0) \circ ('mod' 2) \circ \text{fst}) \$ \text{ zip } [1..] \text{ rs}$ | [1,2,3,4] |
| (g) $\text{head} \circ \text{head} \$ \text{ map } (:loop) \text{ rs}$ | 1 |
| (h) $\text{length} \circ \text{take } 4 \$ \text{ filter } (/= 0) \text{ rs}$ | diverge |

8. Consideramos la siguiente representación de los números naturales.

$$\begin{aligned} \text{data Nat} &= \text{Zero} \mid \text{Succ Nat} \\ \text{foldN} &:: (a \rightarrow a) \rightarrow a \rightarrow \text{Nat} \rightarrow a \\ \text{foldN } h \ e \ \text{Zero} &= e \\ \text{foldN } h \ e \ (\text{Succ } n) &= h (\text{foldN } h \ e \ n) \end{aligned}$$

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

- (a) Dado un natural n , la aplicación $\text{foldN } ([+] \text{ []}) \text{ [] } n$ retorna una lista con n listas vacías
- (b) Dados dos naturales n y m , la aplicación $\text{foldN } \text{Succ } m \ n$ retorna el natural que resulta de la suma de n y m
- (c) Dado un natural n , la aplicación $\text{foldN } (\text{Succ} \circ \text{Succ} \circ \text{Succ}) \ \text{Zero } n$ retorna el natural que triplica a n
- (d) Dado un natural n , la aplicación $\text{foldN } (+1) \ 0 \ n$ retorna el entero equivalente

Respuesta: a)

9. Implemente, sin usar recursión, el operador

$$(.*)> :: (a \rightarrow [b]) \rightarrow (b \rightarrow c) \rightarrow (a \rightarrow [c])$$

que dados $f.*>g$ retorna la función equivalente a aplicar g a los elementos de la lista resultante de aplicar f . Por ejemplo, $(\text{replicate } 4.*>\text{show}) 2$ devuelve la lista `["2", "2", "2", "2"]`.

$$f.*>g = \text{map } g \circ f$$

10. Implemente usando *recursión explícita* la función:

$$\text{filterElem} :: \text{Eq } a \Rightarrow [a] \rightarrow (a \rightarrow [a])$$

que dada una lista de elementos comparables devuelve una función que dado un elemento retorna la lista sin ese elemento. Por ejemplo, $\text{filterElem } [1, 2, 3, 4, 5, 2, 3, 4, 3] 3$ devuelve `[1, 2, 4, 5, 2, 4]`.

$$\begin{aligned} \text{filterElem } [] &= \text{const } [] \\ \text{filterElem } (x : xs) &= \lambda e \rightarrow \text{if } x == e \text{ then } rs \text{ e else } x : rs \text{ e} \\ &\quad \text{where } rs = \text{filterElem } xs \end{aligned}$$

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

$$\text{filterElem} = \text{foldr } (\lambda x \text{ rs } e \rightarrow \text{if } x == e \text{ then } rs \text{ e else } x : rs \text{ e}) (\text{const } [])$$