Ejercicios de Programación Declarativa

Curso 2019/20

Hoja 1

1. Escribe el tipo de las siguientes expresiones, siempre que sea posible. Escribe las que sean sintácticamente correctas en notación simplificada, sin utilizar la constructora de listas (:).

   a. `[ True : [ ]]` b. `[ ] : [True ]`
   c. `[ True ]: [ ]` d. `True : [ True ]`
   e. `1 : (2 : 3 : [ ])` f. `[1 : [2]] : [ [ ] ]`
   g. `[1, 1] : (2 : [ ])` h. `[ ] : [ [ ] ] : [ ]`

2. Escribe el tipo de las siguientes expresiones, siempre que sea posible. Indica cuáles están mal tipadas y por qué.

   a) `head ['a', 'f']`

   b) `tail ['a', 'f']`

   c) `tail head ''af''`

   d) `head (tail ''af'')`

   e) `splitAt 4 ['a' .. 'f']`

   f) `zip [3 + 2, 0] [''af'']`

   g) `drop (+2) [1,2,3]`

   h) `drop (div 2 0) [1,2,3]`

   i_1) `'ab' ++ 'bc'` i_2) `''ab'' ++ ''bc''` i_3) `''ab'' + ''bc''` i_4) `''ab'' ++ 'c'`

3. Determina el valor de las expresiones evaluables del ejercicio anterior.

4. Empareja cada una de las expresiones de la columna izquierda con su equivalente de la derecha:

   1. `0:2:[4]` a. `[[[0],[2,4]]]`
   2. `[0]:([2:(4:[])])` b. `[0]:(2:[4]):[[]]`
   3. `[[0]:(2:[4]):[]]` c. `[0]:([2:(4:[])])`
   4. `[0]:(2:4:[]):[[]]` d. `0:(2:(4:[]))`

5. Encuentra si es posible el valor de las siguientes expresiones y explica por qué no es posible en las que no se pueda.

   a) `let x = y + 1 in let z = x ^ 2 in z`

   b) `let y = let x = 2 in (let z = x ^ 2 in z) in y`

   c) `let y = let x = 2 in (let z = x ^ 2 in z) in z + y`

   d) `let {x = 5; y = 4} in if x < y then x else y`

e) `let {x = 5; y = 4} in if x < y then z = x else z = y`

f) `if [1] !! 1 == 1 then [1] else [ ]`

g) `let x = elem 1 [1] in if x then [1] else [ ]`

h) `let x = elem 1 [ ] in if x then [1] !! 1 else [1] !! 0`

i) `let x = elem 1 [ ] in if x then 1 else [ ]`

6. Indica razonadamente cuáles de los siguientes tipos son equivalentes:

   $\tau_1 =$ `(a ->b) ->(a ->a ->b)`

   $\tau_2 =$ `a ->b ->((a ->a) ->b)`

   $\tau_3 =$ `a ->b ->(a ->(a ->b))`

   $\tau_4 =$ `a ->(b ->(a ->a ->b))`

7. Supuesto que `!` es un operador que se ha declarado como infijo y que asocia por la izquiera (`infixl 4 !`) ¿Cuáles de las siguientes expresiones son sintácticamente correctas? Usa paréntesis para comprobarlo. Transforma e_1 en una expresión equivalente en notación prefija.

   $e_1 =$ `f x y ! g x ! h y`

   $e_2 =$ `((!) (f x y) g x) ! h y`

   $e_3 =$ `(!) ((!) (f x y) (g x)) h y`

8. La función `f` está definida mediante la ecuación:

   ```
   f (x, y, z) = let m = min (min x y) z in m
   ```

   a) ¿Qué calcula `f (x, y, z)`?

   b) Halla el tipo de `f`.

   c) Redefine la función `f` en notación currificada y escribe su tipo.

9. Define una función (sin olvidar declarar su tipo cualificado) que dados tres argumentos, que admitan un orden entre ellos, devuelva una terna compuesta por los tres argumentos en orden creciente.

   a) Usando expresiones `if`.

   b) Usando guardas.

10. Define una función, usando ajuste de patrones, que aplicada a una lista cualquiera dé como resultado `True` si la lista tiene exactamente dos elementos y `False` en caso contrario.

11. Considera el siguiente programa:

    ```
    p :: Int ->Bool
    p n = if n == 0 then True else i (n-1)
    i :: Int ->Bool
    i n = if n == 0 then False else p (n-1)
    ```

    Explica el significado de las funciones `i` y `p`. Indica el valor de la expresión `i 4 || p 4`.