Programación Funcional - Práctico 2

1. Dada la siguiente función

   $$dup\ x = (x, x)$$

   Explique en que difieren $(dup \circ dup)$ y $(dup\ dup)$.

2. Dada la función *twice*

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

   Explique el resultado de hacer $twice\ tail\ [1, 2, 3, 4]$. ¿Es posible hacer $twice\ head\ [1, 2, 3, 4]$? Justifique.

3. Sea $h\ x\ y = f\ (g\ x\ y)$. ¿Cuáles de las siguientes afirmaciones son correctas?

   (a) $h \equiv f \circ g$

   (b) $h\ x \equiv f \circ g\ x$

   (c) $h\ x\ y \equiv (f \circ g)\ x\ y$

4. Implemente usando *pattern matching* una función *sumaPrimeros*, que dada una lista de enteros agrega al principio el resultado de sumar sus dos primeros elementos (si tiene). Por ejemplo $sumaPrimeros\ [1, 2, 3, 4]$ resulta en $[3, 1, 2, 3, 4]$, mientras que $sumaPrimeros\ [1]$ resulta en $[1]$.

5. La función *flip* tiene el siguiente tipo: $(a \to b \to c) \to b \to a \to c$. Observando el tipo, ¿puede determinar qué hace la función?

   (a) Implemente la función *flip*.

   (b) Defina una expresión lambda equivalente a la función *flip*.

6. Usando secciones y composición de funciones, implemente una función $cuentas :: Integer \to Integer$, que dado un número, le sume 3, al resultado lo multiplique por 2, luego le reste 8 y finalmente lo divida por dos.

7. (a) Implemente la función *map* usando listas por comprensión.

   (b) Implemente la función *filter* usando listas por comprensión.

8. Usando *map*, defina una función $squares :: [Int] \to [Int]$ que dada una lista de enteros retorne una lista con los cuadrados de los elementos de la lista.

9. Defina la función *length* en términos de *map* y *sum*.

10. Usando *filter*, defina:

    (a) Una función $all :: (a \to Bool) \to [a] \to Bool$ que dada una condición y una lista, verifique si todos los elementos de la lista cumplen con dicha condición. Ejemplos:
    $all\ (>0)\ [1,2,3]$ retorna *True*
    $all\ (\equiv \texttt{'a'})\ [\texttt{'a'}, \texttt{'b'}, \texttt{'c'}]$ retorna *False*

    (b) Una función $elem :: Eq\ a \Rightarrow a \to [a] \to Bool$ que determina si un elemento pertenece a una lista. Ejemplos:
    $elem\ 2\ [1,2,3]$ retorna *True*
    $elem\ \texttt{'a'}\ [\texttt{'b'}, \texttt{'c'}]$ retorna *False*

11. Indique el tipo y explique lo que hace la siguiente función:

$$rara\ p = filter\ p \circ filter\ (not \circ p)$$

12. Indique el tipo y explique lo que hace la siguiente función:

$$rara2 = zipWith\ (\circ)\ [length, sum]\ [drop\ 4, take\ 4]$$

    Muestre un ejemplo de aplicación correcta de la expresión $(head\ rara2)$ y su resultado.

13. (a) Utilizando *flip*, *mod*, *length*, *map* y *filter*, defina una función que dada una lista de enteros retorne la cantidad de elementos pares que tiene la lista.

    (b) Haga lo mismo, pero sin usar *map*.

    (c) Haga lo mismo, pero sin usar *flip*.

14. La función *filter* se puede definir en términos de *concat* y *map*:

$$\begin{aligned} &filter\ p = concat \circ map\ box \\ &\qquad \textbf{where}\ box\ x = ... \end{aligned}$$

    Dar la definición de *box*.

15. Considere el tipo *Triangulo* definido en el Ejercicio 11 del Práctico 1.

$$\textbf{data}\ Triangulo = Equi\ Int \mid Iso\ Int\ Int \mid Esca\ Int\ Int\ Int$$

    Defina una función $isos :: [Triangulo] \to Int$, que dada una lista de triángulos retorna la cantidad de ellos que son isósceles. Defina *isos* usando:

(a) listas por comprensión.

(b) *filter*

16. Considere la siguiente representación de matrices de dos dimensiones en términos de listas de listas:

    type *Matriz* $a = [[a]]$

    donde vamos a asumir que todas las filas (dadas por las listas de tipo [a]) son del mismo tamaño. Por ejemplo,

    $m = [[1, 2, 3], [4, 5, 6]]$

    representa una matriz de 2x3.

    (a) Usando *drop*, defina una función *columna* :: *Int* $\rightarrow$ *Matriz* $a \rightarrow [a]$ tal que *columna* i m retorna la i-ésima columna de la matriz m.

    (b) Usando *columna*, defina una función *transpose*::*Matriz* $a \rightarrow$ *Matriz* a que transpone una matriz.
    Por ejemplo, *transpose* m retorna $[[1, 4], [2, 5], [3, 6]]$

17. Explique por qué la siguiente definición no es aceptada por el sistema de tipos de Haskell:

    dobleAp $f = (f\ True, f\ \texttt{'a'})$