Lenguajes de programación 2019-1
Práctica 0

Fecha de entrega: 29 de agosto del 2018.

El objetivo de esta práctica es que aprendas a programar (o a reforzar tus conocimientos) en Haskell.

Naturales y Enteros

Considera el siguiente tipo que representa a los números naturales:

```
data N = Cero | S N
```

1. Haz que el tipo `N` forme parte de la clase *Eq*.
2. Haz que el tipo `N` forme parte de la clase *Show* de tal manera que pinte en pantalla los números así:

```
Cero = 0_N
S Cero = 1_N
S (S Cero) = 2_N
⋮
```

3. Implementa las siguientes funciones:
   - `creaN::Int→N`
   - `sumN::N→N→N`
   - `prodN::N→N→N`
   - `menorN::N→N→Bool`
4. Dados $a, b, c, d \in \mathbb{N}$, decimos que $(a, b) \backsim (b, c)$ si y sólo si $a + d = b + c$. Implementa la función `equivR::(N,N)→(N,N)→Bool`.
   Como recordarás de tu curso de Álgebra Superior II, los números enteros se definen como el cociente:

   $$\mathbb{Z} := \mathbb{N} \times \mathbb{N} / \backsim$$

   Así, los números enteros se pueden ver como pares de naturales.
   De esta manera, se define el tipo para representar números enteros:

   ```
   data Z = Clase (N,N) deriving Show
   ```

5. Haz que el tipo `Z` forme parte de la clase *Eq*.
   Para sumar y multiplicar números enteros se hace lo siguiente:

   $$\overline{(a,b)} + \overline{(c,d)} = \overline{(a+c, b+d)}$$
   $$\overline{(a,b)} * \overline{(c,d)} = \overline{(a*c+b*d, a*d+b*c)}$$

   Además:
   - $\overline{(a,b)} = 0$ sii $a = b$
   - $\overline{(a,b)}$ es positivo sii $a > b$
   - $\overline{(a,b)}$ es negativo sii $a < b$
6. Implementa las siguientes funciones:
   - `creaZ::Int→Z`
   - `sumZ::Z→Z→Z`
   - `prodZ::Z→Z→Z`

- `ceroZ::Z→Bool`
- `positivoZ::Z→Bool`
- `negativoZ::Z→Bool`

Listas

Implementa las siguientes funciones:

7. Implementa la función `pyu::[a]->(a,a)` que toma una lista y devuelve los elementos *primero y último* en una tupla.
8. Implementa la función `clona::[Int]->[Int]` que hace lo siguiente: cada que ve un elemento k, repite k-veces la ocurrencia de k en la lista de forma consecutiva.
   Ejemplos:
   - clona [1,2,3] = [1,2,2,3,3,3]
   - clona [4,0,5] = [4,4,4,4,5,5,5,5,5]
   - clona [-2,3,1] = [3,3,3,1]

   Si `clona` ve el cero ó un entero negativo lo borra de la lista.
9. Implementa la función `agrupa::[Int]->[[Int]]` que agrupa los elementos repetidos de manera consecutiva en una lista conforme los va viendo, y devuelve las agrupaciones en una lista. Por ejemplo, agrupa [1,1,1,2,3,3,1] = [[1,1,1],[2],[3,3],[1]]
10. Implementa la función `frec::[Int]->[(Int,Int)]` que hace lo siguiente: dada una lista l devuelve la lista:

$$\{(x, y) \mid x \in l,\ y = \text{el número de veces que aparece } x \text{ en } l\}$$

11. Implementa la función `primos::Int->[Int]` que devuelve la lista de todos los números primos menores que n.

Lectura I

12. Entrega un resumen de cuatro cuartillas de la Parte I del artículo "Hudak, P., Hughes, J., Peyton Jones, S., & Wadler, P. (2007, June). *A history of Haskell: being lazy with class*. In Proceedings of the third ACM SIGPLAN conference on History of programming languages (pp. 12-1). ACM.".