Lógica Computacional
Práctica 1: (Re)Familiarización con Haskell

César Hernández Cruz
Víctor Zamora Gutiérrez
Diego Carrillo Verduzco

8 de agosto de 2018

1. Propósito

El propósito de esta práctica es familiarizarse con la programación funcional en Haskell, utilizando los elementos básicos del lenguaje que se utilizarán a lo largo del curso en muchas de las prácticas.

2. Ejercicios

Definir las siguientes funciones en Haskell, en un archivo nombrado `practica1.hs`

- `remove :: Eq a =>a ->[a] ->[a]`, función que recibe un elemento del tipo `a` y una lista de elementos de tipo `a` y remueve todas las ocurrencias de ese elemento en la lista. Ejemplo:
  ```
  >rm 2 [1,2,3]
  [1,3]
  ```

- `(\\) :: Eq a =>[a] ->[a] ->[a]`, función que recibe dos listas y calcula su diferencia en el sentido conjuntista; esto es, devuelve la primera lista pero habiendo quitado de ella todos los elementos de la segunda lista. Ejemplo:
  ```
  >[1,2,3,4,5]\\[3,2,1]
  [4,5]
  ```

- `rmdup :: Eq a =>[a] ->[a]`, función que remueve todos los elementos duplicados de una lista. Ejemplo:
  ```
  >rmdup [3,2,1,1,2,2,4,5,3,2,1,6]
  [3,2,1,4,5,6]
  ```

- Definir un tipo de dato `Tree a` que represente un árbol binario genérico. Hacer que este tipo de dato derive la clase `Show`.

- `insert :: Ord a =>a ->Tree a ->Tree a`, función que inserta un elemento de un tipo que cumpla `Ord` (es decir, un orden total) en un árbol de ese tipo, de tal forma que si el arbol de entrada es un árbol ordenado, también lo será el árbol de salida. Ejemplo:
  ```
  >insert 0 Empty
  Node 0 Empty Empty
  >insert 1 (Node 0 Empty Empty)
  Node 0 (Node 1 Empty Empty) Empty
  >insert -1 (insert 1 (insert 0 Empty))
  Node 0 (Node 1 Empty Empty) (Node -1 Empty Empty)
  ```

- `fromList :: Ord a =>[a] ->Tree a`, función que toma una lista de elementos ordenables y construye el árbol binario **ordenado** que los contiene (usar `insert`). Ejemplo:
  ```
  >fromList [0,-1,1]
  Node 0 (Node 1 Empty Empty) (Node -1 Empty Empty)
  ```

- `inOrder :: Tree a ->[a]`, función que toma un árbol y devuelve una lista con los elementos del árbol ordenados según el recorrido *in order* de éste. (Recordar que el recorrido *in order* de un árbol primero visita el subárbol izquierdo del nodo actual, luego el elemento del nodo actual y luego el subárbol derecho). Ejemplo:
  ```
  >inOrder (Node 2 (Node 1 (Node 0 Empty Empty) Empty) (Node 3
  Empty Empty))
  [0, 1, 2, 3]
  ```

- `sort :: Ord a =>[a] ->[a]`, función que toma una lista de elementos ordenables y devuelve una lista de esos elementos ordenados. **PROTIP:** Si `fromList` tiene tipo `[a] ->Tree a` e `inOrder` tiene tipo `Tree a ->[a]` entonces componer estas dos funciones tiene tipo `[a] ->[a]`. Ejemplo:

```
>sort [5,4,10,21,-3,0,-15]
[-15,-3,0,4,5,10,21]
```

3. Formato de entrega

La práctica se entregará de la siguiente manera: todos los archivos relacionados con la práctica deben estar en una carpeta nombrada `PracticaN` donde N es el número de la práctica. En esta carpeta, además de los archivos de la práctica, debe haber un archivo nombrado simplemente `README` que contenga el nombre y número de cuenta del alumno que entrega, en ese orden, en dos lineas. Ejemplo:

```
Diego Carrillo Verduzco
316942069
```

Comprimir esta carpeta en alguno de los formatos de compresión comunes: `.tar.gz., .zip, .rar` (de preferencia `.tar.gz`) y subir la carpeta comprimida como solución a la tarea en el Google Classroom.