Práctica Tipos - BT - BST - RBT - Leftist Heaps

1. Dado el siguiente tipo de dato que representa los números naturales:

data Nat = Cero | Succ Nat

a) ¿Qué tipo tiene Succ?

b) Definir la función *int2Nat* :: Int → Nat que dado un entero retorne su representaciń en Nat
Ejemplo *int2Nat* 4 = Succ (Succ (Succ (Succ Cero)))

c) Definir la función *suma* :: Nat → Nat → Nat
NO convertir los Nat a enteros para poder sumarlos.

d) Definir la función *nat2Int* :: Nat → Int que dado un Nat retorne a que entero representa.

2. Dado el tipo de datos de árboles binarios:

data Arb = E | H Int | N Arb Arb

y el tipo de datos de comandos, para navegar el árbol:

data Cmd = L | R

a) ¿Qué tipo tiene N?

b) Definir la función parcial *selec*::[Cmd] → Arb → Arb, que selecciona el subárbol correspondiente. Por ej:

```
                  /\
select [L,R] ( /  \ ) = Hoja 4
              /\   5
             /  \
            3   4
```

La función *selec* es parcial, ya que no está definida para listas de comandos erróneas (como por ejemplo, ir a la izquierda cuando nos encontramos en una hoja).

c) Definir una función *enum* :: Arb → [[Cmd]] que devuelva todas las secuencias de comandos válidas para ir desde la raíz hasta una hoja.

3. Un lenguaje imperativo simple sólo permites variables de un único tipo, para esto se mantiene un estado con el nombre de las variables y sus valores. Un Estado es una estructura secuencial formada por un nombre de variable y el valor correspondiente. Se requiere las siguientes operaciones sobre Estado :

inicial : Estado a
update : Nombre → A → Estado a → Estado a
lookfor : Nombre → Estado a → Maybe a
free : Nombre → Estado a → Estado a

donde

- *inicial* representa el estado inicial de un programa donde no sean definido ninguna variable
- *update* permite actualizar el valor de una variable existente y si la variable no existe la agrega al estado con el valor dado.
- *lookfor* dado el nombre de una variable permite obtener el valor de esta si es que existe en el estado.
- *free* dado el nombre de una variable la elimina del estado.

Definir los tipo de dato para Nombre y Estado e implementar las operaciones dadas.

4. Implementar una función que:

a) calcule el número de nodos en un nivel específico de un árbol binario

b) reciba un árbol binario de búsqueda y verifique si es un árbol balanceado, es decir, que la diferencia de alturas entre su subárbol izquierdo y derecho no sea mayor que 1 para todos los nodos

c) encuentren el sucesor y el predecesor de un valor dado en un árbol binario de búsqueda. El sucesor es el valor más pequeño mayor que el valor dado, y el predecesor es el valor más grande menor que el valor dado

d) dado un Leftist Heaps, retorne una lista con sus elementos ordenados de mayor a menor

e) verifique si un árbol cumple con la propiedad de Leftist Heap

f) elimine todos los elementos duplicados en un Leftist Heap y devuelva el nuevo heap resultante

g) verifique si un árbol cumple con la propiedad de $\mathsf{Red} - \mathsf{Black} - \mathsf{Tree}$