1er Parcial – 2020s2 - Estructuras de Datos – UNQ

Aclaraciones:

- *Esta evaluación es a libro abierto. Se pueden usar todas las funciones y tipos de datos vistos en la práctica y en la teórica, salvo que el enunciado indique lo contrario.*
- *Recuerde respetar el formato del mail con el que se entrega la solución al examen.*
- *Respire hondo, tome pausas cuando corresponda, y recuerde que de cometer errores se aprende.*

La fiambrería

Los siguientes tipos de datos permiten representar los sanguches de una fiambrería:

```
data Sanguche = Pan Relleno
data Relleno = Feta TipoDeFeta Relleno
             | Aire
data TipoDeFeta = Queso | Jamon | Mortadela | Salame
```

Resolver las siguientes funciones utilizando *recursión estructural*:

a) `rellenoDeAire :: Sanguche -> Bool`
Propósito: Dado un sanguche, indica si el relleno es solo de aire.

b) `esTortitaDeJamon :: Sanguche -> Bool`
Propósito: Dado un sanguche indica si solo tiene fetas de jamon.

c) `mandaleNDe :: Int -> TipoDeFeta -> Sanguche -> Sanguche`
Propósito: Dados un número `n` y un tipo de feta, agrega `n` fetas de ese tipo, al principio del relleno del sanguche dado.

d) `peroSinQueso :: Sanguche -> Sanguche`
Propósito: Quita todo el queso del relleno al sanguche dado.

e) `ordenadosPorCantidad :: Sanguche -> [(TipoDeFeta, Int)]`
Propósito: Devuelve una lista de fetas del relleno del sanguche, junto con su cantidad de apariciones.

El Laberinto

Los siguientes tipos de datos permiten representar un laberinto con bifurcaciones, cofres cerrados por llaves y salidas:

```
-- Las llaves son simples números.
type Llave = Int

-- Existen direcciones que indican caminos dentro del laberinto.
data Dir = Izq | Der

-- Los cofres tienen una cantidad de oro,
-- pero solo accesible para aquellos que posean las llaves que lo abren.
-- Pueden existir cofres que no requieran de llaves.
data Cofre = Cofre [Llave] Int

-- Un laberinto posee salidas, celdas y bifurcaciones.
-- En las celdas hay cofres, en las salidas no hay nada,
-- y en las bifurcaciones puedo ir hacia uno u otro lado.
data Laberinto = Salida
               | Celda Cofre
               | Bifurcacion Laberinto Laberinto
```

Resolver las siguientes funciones utilizando *recursión estructural*:

a) `cantidadDeSalidas :: Laberinto -> Int`
Propósito: Indica cuantas salidas posee un laberinto.

b) `queLlavesDeboTener :: Laberinto -> [Llave]`
Propósito: Dado un laberinto indica qué llaves se deben tener para poder abrir todos sus cofres.
Nota: el resultado no debe tener llaves repetidas.

c) `cantidadDeOroCon :: [Llave] -> Laberinto -> Int`
Propósito: Indica cuánto oro puede conseguirse dada una lista de llaves.

d) `haySalidaPor :: [Dir] -> Laberinto -> Bool`
Propósito: Dada una lista de direcciones indica si llevan a una posible salida del laberinto.

e) `salidaMasCercana :: Laberinto -> [Dir]`
Propósito: Indica el camino a la salida más cercana.
Precondición: Existe al menos una salida.