Práctica 1
Lógica computacional 2020-2
Repaso de Haskell
Fecha de entrega: **domingo 16 de febrero de 2020**

1. Dados dos puntos en el plano, realiza una función que encuentre el punto medio entre ellos dos.

   Firma de la función:
   ```
   puntoMedio :: (Float,Float) -> (Float,Float) -> (Float,Float)
   ```

   Ejemplo:
   ```
   Prelude>puntoMedio (-1,2) (7,6)
   (3.0,4.0)
   ```

2. Los números complejos los podemos entender como una pareja ordenada de puntos en el plano (a, b), la cual representa al número $a + ib$, donde a es la parte real y b es la parte imaginaria o compleja. Si consideramos a esta pareja ordenada como números complejos, tenemos una pareja ordenada **Complejo = (Float,Float)**; bajo esta premisa, realice la siguiente función sobre números complejos.

   Las ecuaciones cuadráticas $ax^2 + bx + c = 0$ tienen dos soluciones reales, o dos complejas, realiza una función que obtenga las soluciones de esta ecuación ya sean reales o complejas.

   Firma de la función:
   ```
   raices :: Float -> Float -> Float -> (Complejo,Complejo)
   ```

   Ejemplo:
   ```
   Prelude>raices 5 9 3
   ((-0.44174242,0),(-1.3582575,0))
   ```

3. Definir una función que dada una lista y dos enteros, regrese la lista de los elementos comprendidos entre las posiciones m y n.

   Firma de la función:
   ```
   segmento :: Int -> Int -> [a] -> [a]
   ```

   Ejemplo:
   ```
   segmento 3 4 [3,4,1,2,7,9,0]
   [1,2]
   segmento 3 5 [3,4,1,2,7,9,0]
   [1,2,7]
   segmento 5 3 [3,4,1,2,7,9,0]
   []
   ```

4. Definir una función que dada una lista y un número, entregue una lista formada por los n primeros elementos de la lista y los n elementos finales.

   Firma de la función:
   ```
   extremos :: Int -> [a] -> [a]
   ```

   Ejemplo:
   ```
   extremos 3 [2,6,7,1,2,4,5,8,9,2,3]
   [2,6,7,9,2,3]
   ```

5. Define una función que elimine un intervalo de una lista. Dados dos números y una lista elimina los elementos que se encuentran en el intervalo de esos dos números.

   Firma de la función:
   ```
   dIntervalos :: Int -> Int -> [a] -> [a]
   ```

   Ejemplo:
   ```
   dIntervalos 2 4 [1,2,3,4,5,6,7]
   [1,5,6,7]
   ```

6. Un número natural n se denomina abundante si es menor que la suma de sus divisores propios, exceptuando el propio número. Por ejemplo, 12 y 30 son abundantes pero 5 y 28 no lo son. Definir una función tal que dado un número natural, regrese la lista de números abundantes menores o iguales que él.

   Firma de la función:
   ```
   numerosAbundantes :: Int -> [Int]
   ```

   Ejemplo:
   ```
   numerosAbundantes 50
   [12,18,20,24,30,36,40,42,48]
   ```

   Recursión

7. Definir una función que dada una lista y regrese una lista tal que contiene los mismos elementos que la original pero sin duplicados.

   Firma de la función:
   ```
   eliminaDuplicados :: Eq a => [a] -> [a]
   ```

   Ejemplo:
   ```
   eliminaDuplicados [1,3,1,2,3,2,1]
   [1,3,2]
   ```

8. Se define el primitivo de un número como sigue:

   Dado un número natural n, multiplicamos todos sus dígitos, repetimos este procedimiento hasta que resulte un solo dígito al cual llamamos primitivo de n. Por ejemplo, para 327; 327: $3 \times 2 \times 7 = 42$ y $4 \times 2 = 8$. Por lo tanto, el primitivo de 327 es 8.

   Definir la función dado un número nos regrese el primitivo.

   Firma de la función:
   ```
   primitivo :: Integer -> Integer
   ```

   Ejemplo:
   ```
   primitivo 327
   8
   ```

   Las siguientes líneas de código sirven para definir los tipos de números naturales, booleanos, listas y árboles de manera recursiva.

```
data Nat = Cero | Suc Nat deriving(Show,Eq)

data Lista a = Nula | Cons a (Lista a) deriving(Show,Eq)

data Arbol = Vacio | Nodo Arbol Int Arbol deriving(Show,Eq)
```

Considerando estas definiciones, define las siguientes funciones.

9. Definir una función que dadas dos Listas y un número Natural j, regresa una lista tal que, se encuentran concatenados el i-ésimo elemento de la primer Lista con el i-ésimo elemento de la segunda Lista; a partir del elemento j de cada una de las listas.

   Firma de la función:
   `sipLis :: Nat -> Lista a -> Lista a -> Lista a`

   Ejemplo:
   `sipLis (Suc (Suc Cero)) (Cons 2 (Cons 1 Nula)) (Cons 3 (Cons 5 (Cons 7 Nula)))`
   `Cons 1 (Cons 5 Nula)`

10. Definir una función la cual convierte un árbol en una lista haciendo el recorrido en preorden.

    Firma de la función:
    `aplanaArbolPre :: Arbol -> [Int]`

    Ejemplo:
    `aplanaArbolPre arbol1`
    `[2,0,-1,1,4,3,5]`.

    Donde arbol1 es:
    `(Nodo (Nodo (Nodo Vacio (-1) Vacio) 0 (Nodo Vacio 1 Vacio)) 2 (Nodo (Nodo Vacio 3 Vacio) 4 (Nodo Vacio 5 Vacio)))`

NOTA

Recuerden que deben seguir los lineamientos establecidos para la entrega de su práctica, los cuales se encuentran en la página del curso.