Música Funcional

Parcial de Paradigmas de Programación - Paradigma Funcional - Jueves Noche - 2021

Nota: Leé TODO el enunciado antes de empezar, y no te olvides de las funciones que vas programando: seguramente te van a servir más adelante!

La compañía detrás de un conocido reproductor de música online nos encarga modelar un sistema que permita pre-procesar y analizar **Canciones** para mejorar su plataforma.

Decimos que una **Canción** es una secuencia de **Notas**. De cada **Nota** conocemos su **tono** (la frecuencia a la que suena), su **volumen** (la intensidad con la que suena) y su **duración** (el tiempo que se mantiene sonando).
Para representar estos datos contamos con las siguientes definiciones:

```haskell
type Cancion = [Nota]

data Nota = Nota {
  tono :: Float,     -- Frecuencia medida en Hz
  volumen :: Float, -- Volumen de reproducción medido en Db
  duracion :: Float -- Tiempo de reproducción medido en segundos
} deriving (Eq, Show)

-- FUNCIONES AUXILIARES

cambiarVolumen :: (Float -> Float) -> Nota -> Nota
-- Dada una función transformación y una nota retorna una copia de la
-- nota con el volumen igual al resultado de aplicar la transformación a
-- su volumen actual.
cambiarVolumen delta nota = nota { volumen = delta (volumen nota) }

cambiarTono :: (Float -> Float) -> Nota -> Nota
-- Dada una función transformación y una nota retorna una copia de la
-- nota con el tono igual al resultado de aplicar la transformación a
-- su tono actual.
cambiarTono delta nota = nota { tono = delta (tono nota) }

cambiarDuracion :: (Float -> Float) -> Nota -> Nota
-- Dada una función transformación y una nota retorna una copia de la
-- nota con la duración igual al resultado de aplicar la transformación a
-- su duración actual.
cambiarDuracion delta nota = nota { duracion = delta (duracion nota) }
```

```haskell
promedio :: [Float] -> Float
-- Dada una lista de números retorna el valor promedio
promedio lista = sum lista / fromIntegral (length lista)
```

Teniendo en cuenta este dominio se pide utilizar las herramientas y conceptos provistos por el Paradigma Funcional para modelar los puntos descritos a continuación. Recuerden que todas las definiciones de funciones deben estar acompañadas de su tipo.

1) Definir las funciones necesarias para identificar cuando una **Nota**…

a) **esAudible**: Decimos que una nota es “audible” si su tono está entre 20Hz y 20.000Hz y su volumen es mayor a 10Db.

b) **esMolesta**: Una nota se considera molesta si su tono y volumen están dentro de ciertos rangos:

i) Las notas audibles con un tono menor a los 250Hz son molestas si su volumen es mayor a los 85Db.

ii) Las notas audibles con un tono mayor o igual a los 250Hz son molestas si su volumen es mayor a los 55Db.

iii) Las notas no-audibles nunca resultan molestas.

2) Dada una **Cancion**, implementar las siguientes funciones de análisis:

a) **silencioTotal**: Retorna la sumatoria de las duraciones de todas las notas de la canción que no son audibles.

b) **sinInterrupciones**: Se cumple cuando todas las notas de duración mayor a 0.1 segundos son audibles.

c) **peorMomento**: Retorna el máximo volumen al que se reproduce una nota molesta de la canción.

3) No alcanza con encontrar problemas en la música; también queremos corregirlos! Para eso vamos a modelar **Filtros**. Podemos pensar en un **Filtro** como una herramienta que nos permite cambiar una **Canción** y ajustar sus notas de acuerdo a algún criterio.

Se pide definir el tipo de dato **Filtro** junto con cualquier lógica que permita modelar los siguientes **Filtros**:

Nota: No se olviden de las funciones auxiliares que se dan al principio! Podrían resultar muy útiles…

a) **trasponer**: Dado un escalar (o sea, un número), este filtro modifica cada nota de una canción, multiplicando su tono por el escalar.

b) **acotarVolumen**: Dado un volúmen máximo y un volúmen mínimo, este filtro modifica cada nota de una canción, subiendo el volumen de aquella con volumen menor al mínimo y bajando el de aquellas con volumen mayor al máximo.

c) **normalizar**: Este filtro modifica cada nota de una canción seteando su volumen al volumen promedio de la canción original.

4) Dada la función **f**:

```
f g [] y z = g y z
f g (x : xs) y z = g (x y) z || f g xs y z
```

Se pide:

a) Identificar su tipo

b) Utilizar **f** para definir la función:

```
infringeCopyright :: [Filtro] -> Cancion -> Cancion -> Bool
```

Que dada una lista de filtros, una canción original y una canción sospechosa indica si nos encontramos ante una copia. Esto es así cuando la canción sospechosa es igual a la canción original o si alguno de los filtros dados, al ser aplicado sobre la canción original produce una canción igual a la copia.

c) ¿Qué te parece f en términos de Expresividad y Declaratividad? ¿Se puede mejorar alguno de esos aspectos? ¿Cómo? Justificar la respuesta.

5) Definir una función **tunear** que dada una **lista de Filtros** y una **Canción** retorne la **Canción** resultante de aplicarle todos los filtros a la canción original y, por último, la **normalice**.