Programación Funcional

Aclaraciones:

- *Esta evaluación es a libro abierto. Se pueden usar todas las funciones y propiedades vistas en clase, aclarando la referencia. Cualquier otra función o propiedad que se utilice* ***debe ser definida o demostrada****.*
- *No se olvide de poner nombre, nro. de alumno, nro. de hoja y cantidad total de hojas en cada una de las hojas.*
- *Le recomendamos leer el enunciado en su totalidad y organizar sus ideas antes de comenzar la resolución.*
- *Recuerde que reusar código es una forma muy eficiente de disminuir el tiempo necesario para programar.*
- *La intención de la evaluación es medir cuánto comprende usted del tema. Por ello, no dude en escribir todo lo que sabe, explicar lo que se propone antes de escribir código y probar sus funciones con ejemplos.*

Considere la siguiente representación de composiciones musicales:

```
data Nota = Do | Re | Mi | Fa | Sol | La | Si deriving (Eq, Ord)

type Tiempo   = Int  -- el instante en el que suena una nota

type Duracion = Int  -- cantidad de tiempos que suena una nota

data Comp = Silencio Duracion      |
            Batido   Nota Duracion |
            Arpegio  Comp Comp     | -- composiciones en secuencia
            Acorde   Comp Comp       -- composiciones en paralelo
```

Y considere también los siguientes datos utilizados por interfaces de reproducción de audio:

```
type Midi = [[Nota]] -- notas simultáneas para cada tiempo en la duración de la melodía

type Sampler = Tiempo -> [Nota] -- función que da las notas que suenan en un tiempo dado
```

Ejercicio 1

Escriba las siguientes funciones:

a) `alargar :: Int -> Comp -> Comp`
que retorna una composición donde la duración cada nota está multiplicada por un factor con respecto a la duración original.

b) `duracion :: Comp -> Int`
que retorna la duración total de la melodía compuesta (considerando que los acordes terminan cuando termina el más largo).

c) `sintetizar :: Comp -> Midi`
que retorna una lista con un elemento por cada tiempo de la duración de la melodía compuesta. Cada elemento de dicha lista es una lista con las notas que suenan simultáneamente en el tiempo indicado (un elemento vacío indica silencio).

Ejercicio 2

Indique el tipo y de una implementación de una función que generalice la recursión sobre estas composiciones (`foldComp`) y escriba las funciones del putno 1 utilizando esta función (sin utilizar recursión explícita).

Ejercicio 3

Escriba las siguientes funciones sin utilizar recursión explícita, es decir, usando los esquemas de recursión de listas o listas por comprensión.

a) `silencios :: Comp -> [Tiempo]`
que retorna todos los tiempos compuestos únicamente por silencio.

b) `unir :: [Comp] -> [Comp] -> Comp`
que dadas dos listas de composiciones de igual longitud retorna una nueva composición donde la melodía de cada posición de una lista suena en simultáneo con la correspondiente en la otra lista. Es decir, la composiciones i-ésimas debe sonar en simultáneo y en secuencia con las demás.

c) `samplear :: Comp -> Sampler`
que dada una composición retorna un sampler en el que la melodía se repite infinitamente.

d) `superponer :: [Sampler] -> Duracion -> Midi`
que retorna un Midi con la duración dada, donde en cada tiempo se escucha la superposición de los samplers en la lista de entrada.

Ejercicio 4

Demostrar la siguiente propiedad:

$$\forall \texttt{c:Comp, k:Int}, k \geq 0 \Rightarrow \texttt{duracion (alargar k c) = k * duracion c}$$