Parcial 1 - Algoritmos I Taller: Tema A

Ejercicio 1

Se va a representar el juego **piedra-papel-tijera** usando tipos en Haskell.

Para ello:

a) definir el tipo `Forma` que consta de los constructores sin parámetros `Piedra`, `Papel` y `Tijera`. **El tipo `Forma` no debe estar en la clase `Eq`**. Luego programar la función

```
le_gana :: Forma -> Forma -> Bool
```

que dadas dos valores `f1` y `f2` de tipo `Forma` devuelve `True` si y sólo si la forma `f1` gana sobre la forma `f2`. La regla es que:

- La piedra le gana a la tijera,
- La tijera le gana al papel
- El papel le gana a la piedra

Se puede seguir la siguiente tabla:

f1	f2	le_gana f1 f2
Piedra	Tijera	True
Piedra	Piedra	False (empate)
Piedra	Papel	False (la forma f1 perdió)
Papel	Piedra	True
Papel	Papel	False (empate)
Papel	Tijera	False (la forma f1 perdió)
Tijera	Papel	True
Tijera	Tijera	False (empate)
Tijera	Piedra	False (la forma f1 perdió)

b) Definir el tipo `Nombre` como un sinónimo de `String`. Luego definir el tipo `Jugador` que consta de un único constructor `Mano` que toma dos parámetros, el primero de tipo `Nombre` y el segundo de tipo `Forma`. Por último programar la función:

```
ganador :: Jugador -> Jugador -> Maybe Nombre
```

que dados dos valores `j1` y `j2` del tipo `Jugador` debe devolver el nombre del jugador ganador (el de `j1` o `j2` según corresponda) usando el constructor `Just`, o `Nothing` en caso que no haya ganador.

Ejercicio 2

Programar la función

```
quien_jugo :: Forma -> [Jugador] -> [Nombre]
```

que dada una forma `f` y una lista de jugadores `js` devuelve los nombres de los jugadores de `js` que usaron la forma `f`

- Inventar un ejemplo concreto con una lista de al menos 3 elementos, ejecutarlo y decirlo como comentario en lo que se suba al parcial.

Ejercicio 3

Definir el tipo `NotaMusical` que con constructores `Do`, `Re`, `Mi`, `Fa`, `Sol`, `La`, `Si`. Los constructores no toman parámetros. Luego definir el tipo `Figura` de constructores `Negra` y `Corchea`. A partir de los tipos anteriores, definir el tipo recursivo `Melodia` cuyos constructores son:

- `Entonar`: Toma tres parámetros. El primero de tipo `NotaMusical` (la nota que se agrega), el segundo de tipo `Figura` (que representa la duración de la nota que se esta agregando) y el tercero de tipo `Melodia` (la melodía a la que se agrega la nueva nota y figura).
- `Vacia`: No toma parámetros y representa la melodía vacia.

Un ejemplo de melodía sería:

```
Entonar Re Negra (Entonar Mi Corchea (Entonar Fa Negra (Entonar Mi Negra Vacia)))
```

Finalmente programar la función:

```
contar_tiempos :: Melodia -> Int
```

que cuenta la cantidad de tiempos que tiene la melodía, sumando `2` por cada figura construida con `Nergra` y sumando `1` para las figuras construidas con Corchea (por lo

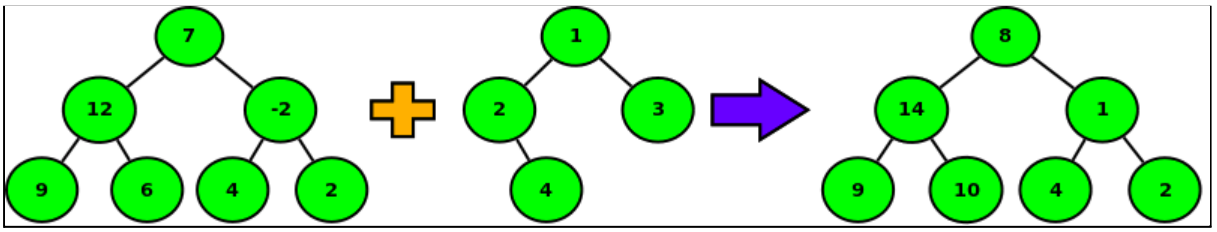
general las negras se consideran un tiempo y las corcheas medio tiempo, pero para evitar decimales se pide de esta otra forma). Si le llamamos `pink` al ejemplo de más arriba, `contar_tiempos pink` debería dar `7`.

Ejercicio 4*

Programar la función

```
arbol_sum :: Arbol Int -> Arbol Int - > Arbol Int
```

que dado dos árboles `as` y `bs` devuelve un nuevo árbol cuyos valores son la suma de los elementos `as` y `bs` punto a punto. Ejemplo:



Además, la suma de árboles es conmutativa, o sea que: