Parcial 1 - Algoritmos I Taller: Tema D

Ejercicio 1

Se van a representar los datos de deportistas. Para ello:

a) Definir:

- El tipo `Deportista` que consta de tres constructores:
  - Constructor `Futbolista`: Tiene tres parámetros, el primero de tipo `Nombre`, el segundo de tipo `Zona` y el tercero de tipo `Titulos`.
  - Constructor `Tenista`: Tiene tres parámetros, el primero de tipo `Nombre`, el segundo de tipo `Categoria` y el tercero de tipo `Titulos`
  - Constructor `Velocista`: Tiene dos parámetros, el primero de tipo `Nombre` y el segundo de tipo `Titulos`.
- El tipo `Nombre`: Debe ser un sinónimo de `String` se usará para los nombres de los deportistas
- El tipo `Zona`: Tiene constructores `Arco`, `Defensa`, `Mediocampo` y `Delantera`, todos constructores sin parámetros.
- El tipo `Categoria`: Tiene constructores `Simples` y `Dobles` ambos sin parámetros.
- El tipo `Titulos`: Debe ser sinónimo del tipo `Int` y se usan para la cantidad de títulos del deportista.

b) Programar la función usando *pattern matching*:

```
misma_zona :: Zona -> Zona -> Bool
```

que dados dos valores `z1` y `z2` del tipo `Zona` debe devolver `True` cuando `z1` y `z2` la misma zona de juego (se construyen con el mismo constructor) y `False` en caso contrario.

Si se usan más de cinco casos, este apartado sumará menos puntaje.

c) Programar la función

```
puntaje_titulos :: Deportista -> Int
```

que devuelve un valor de puntaje según la cantidad de títulos que tiene el deportista. El criterio es el siguiente:

- Si el deportista es tenista: El puntaje será la cantidad de títulos
- Si el deportista es futbolista: El puntaje será el doble de la cantidad de títulos
- Si el deportista es velocista: El puntaje será el triple de la cantidad de títulos.

d) Incluir el tipo `Deportista` en la clase `Ord` de manera tal que un deportista se considere mayor que otro si su valor según la función `puntaje_titulos` es más grande

Ejercicio 2

a) Programar de manera recursiva la función

```
futbolistas_zona :: [Deportista] -> Zona -> [Nombre]
```

que dada una lista de deportistas `ds` y una zona `z` devuelve una lista con los nombres de los los futbolistas de `ds` que juegan en la zona `z`.

b) Escribir una lista de deportistas con al menos tres elementos, donde uno de ellos debe ser un tenista, y otro debe ser un futbolista.

c) Escribir el resultado de `futbolistas_zona` para la lista del punto b)

Ejercicio 3

Basados en el tipo `ListaAsoc` del *Proyecto 2*, programar la función:

```
la_adicionar :: ListaAsoc a b -> b -> ListaAsoc a b
```

que dada una lista de asociaciones `la` y un valor `x` devuelve una nueva lista de asociaciones que resulta de adicionar `x` a cada valor de las asociaciones de `la`. Completar el tipado de la función para incluir los *type classes* necesarios para programarla.

Ejercicio 4*

a) Programar la función

```
a_filter :: (a -> Bool) -> Arbol a -> Arbol (Maybe a)
```

que dado un predicado `p` de y un árbol `as` idevuelve un nuevo árbol con los elementos de `as` aplicándoles el constructor `Just` cuando cumplen con el predicado `p` y reemplazándolos por `Nothing` cuando no satisfacen `p`.

b) Inventar un ejemplo de uso de la función creando un árbol con al menos 3 elementos

c) Escribir el resultado de la función aplicada al ejemplo del inciso b)