Pociones 19

Se pide desarrollar un programa Haskell que ayude a analizar las pociones que se enseñan a los alumnos en el `colegioHogwartsDeMagiaYHechicería`, y los efectos que pueden hacer sobre personas. Las siguientes son algunas funciones ya definidas:

```
aplicar3 f (a, b, c) = (f a, f b, f c)

invertir3 (a, b, c) = (c, b, a)
```

```
suerte (s, _, _) = s
convencimiento (_, c, _) = c
fuerzaFisica (_, _, ff) = ff
```

```
sinRepetidos [] = []
sinRepetidos (x:xs)
  | elem x xs = sinRepetidos xs
  | otherwise = x : sinRepetidos xs
```

```
maximoF _ [ x ] = x
maximoF f ( x : y : xs)
  | f x > f y = maximoF f (x:xs)
  | otherwise = maximoF f (y:xs)
```

Y también tenemos distintos elementos propios de Hogwarts, compuestos de la siguiente manera:

- Cada tupla persona tiene el nombre en su primer componente y otra tupla con tres niveles en el segundo componente: nivel de suerte, de poder de convencimiento, y de fuerza física.
- De las pociones conocemos su nombre y una lista de ingredientes.
- Cada ingrediente tiene un nombre, una cantidad en gramos, y una lista de efectos.
- Un efecto recibe los niveles de una persona y los modifica en cierta forma.

```
harry = ("Harry", (11, 5, 4))
ron = ("Ron", (6, 4, 6))
hermione = ("Hermione", (8, 12, 2))
draco = ("Draco", (7, 9, 6))
```

```
felixFelices = Pocion "Felix Felices" [escarabajosMachacados, ojoDeTigreSucio]
multijugos = Pocion "Multijugos" [cuernoDeBicornioEnPolvo, sanguijuelaHormonal]
floresDeBach = Pocion "Flores de Bach" [orquideaSalvaje, rosita]
```

```
escarabajosMachacados = Ingrediente "Escarabajos Machacados" 52 [f1, f2]
ojoDeTigreSucio = Ingrediente "Ojo de Tigre Sucio" 2 [f3]
cuernoDeBicornioEnPolvo = Ingrediente "Cuerno de Bicornio en Polvo" 10
   [invertir3, (\(a, b, c) -> (a, a, c))]
sanguijuelaHormonal = Ingrediente "Sanguijuela Hormonal" 54 [aplicar3 (*2),
   (\(a, b, c) -> (a, a, c))]
orquideaSalvaje = Ingrediente "Orquídea Salvaje" 8 [f3]
rosita = Ingrediente "Rosita" 1 [f1]
```

```
f1 (ns, nc, nf) = (ns+1, nc+2, nf+3)
f2 = aplicar3 (max 7)
f3 (ns, nc, nf)
  | ns >= 8 = (ns, nc, nf+5)
  | otherwise = (ns, nc, nf-3)
```

A partir de la base de conocimiento presentada, se pide resolver los siguientes puntos utilizando los conceptos aprendidos del paradigma funcional: composición, aplicación parcial y orden superior.

1) Definir los alias de tipo y/o tipos de dato necesarios.

2) Dada una tupla de niveles, definir las funciones:

- **a)** `sumaNiveles`, que suma todos los niveles
- **b)** `diferenciaNiveles`, es la diferencia entre el nivel más alto y el nivel más bajo

3) Dada una tupla persona, definir las funciones:

- **a)** `sumaNivelesPersona`, por ejemplo la suma de niveles de `harry` es 20 (11 + 5 + 4).
- **b)** `diferenciaNivelesPersona`, que aplicada a `harry` debería ser 7 (11 - 4).

4) Definir la función `efectosDePocion`, que recibe una poción y devuelve una lista con todos los efectos de cada uno de sus ingredientes.

```
> efectosDePocion felixFelices
[f1,f2,f3] --las funciones no se pueden mostrar, pero es a modo de ejemplo
```

5) Definir la función `pocionesHeavies`, que recibe una lista de pociones y devuelve los nombres de las pociones que tienen al menos 4 efectos.

```
> pocionesHeavies [felixFelices, multijugos, floresDeBach]
["Multijugos"]
```

6)

- **a)** Definir la función `incluyeA` que espera dos listas, devolviendo `True` si la primera está incluida en la segunda.

  ```
  > incluyeA [3, 6, 9] [1..10]
  True
  ```

- **b)** Definir la función `esPocionMagica`. Una poción es mágica si el nombre de alguno de sus ingredientes tiene todas las vocales y además de todos los ingredientes se pide una cantidad de gramos par. Por ejemplo, `multijugos` es mágica, `felixFelices` no lo es porque ningún nombre incluye todas las vocales, `floresDeBach` tampoco porque tiene un ingrediente con cantidad impar de gramos.

7) Definir la función `tomarPocion`, que recibe una poción y una persona, y devuelve una tupla persona que muestra cómo quedaría la persona después de haber tomado la poción. Cuando una persona se toma una poción, se le aplican todos los efectos de cada ingrediente de la poción, en orden.

```
> tomarPocion felixFelices harry
("Harry", (12, 7, 12)) --porque le aplica f1, f2 y por último f3
```

8) Definir la función `esAntidoto`, que recibe una persona y dos pociones, y devuelve `True` en caso de que la segunda poción revierta el efecto de la primera sobre la persona. Es decir, si la persona queda igual después de tomar la primer poción y después la segunda.

9) Definir la función `personaMasAfectada`, que recibe una poción, una ponderación de niveles y una lista de personas, y devuelve la persona que después de tomarse la poción hace máximo el valor de la ponderación de niveles. Una ponderación de niveles es una función que espera una terna de niveles (suerte, convencimiento, fuerza física) y devuelve un número.

10) Escribir consultas que, usando la función del punto anterior, respondan la persona que quedó más afectada según las siguientes ponderaciones:

- **a)** suma de niveles (suerte, poder de convencimiento y fuerza física)
- **b)** promedio de niveles (puede ser el promedio entero)
- **c)** fuerza física
- **d)** diferencia de niveles