Harry Potter y el Examen de Funcional

Nos piden modelar un programa que permita analizar los conflictos entre magos y, obviamente, nuestro primer impulso es hacerlo en funcional. De los magos, sabemos que tienen un nombre, una edad y cierta cantidad de salud. Además, cada mago conoce un conjunto de hechizos el cual puede usar para hacerle cosas raras a otros magos.

Se pide resolver los siguientes puntos, aprovechando al máximo los conceptos del paradigma funcional:

1. Elegir un tipo de dato con el que representar a los Magos y los Hechizos (justificando brevemente la elección) de forma tal que se respete la descripción previa del dominio y sea posible modelar los siguientes hechizos:

    a. **curar**: Este hechizo hace que el mago sobre el que lanzamos el hechizo recupere una cierta cantidad de salud. Este hechizo puede usarse para curar distintas cantidades de vida.

    b. **lanzarRayo**: Este hechizo le hace daño al mago sobre el que se lanza. Si la salud de dicho mago es mayor a 10, le hace 10 puntos de daño, de lo contrario le quita la mitad de su vida actual.

    c. **amnesia**: El mago objetivo olvida los primeros N hechizos que conozca. Se puede lanzar este hechizo con diferentes valores de N.

    d. **confundir**: El mago objetivo se ataca a sí mismo con su primer hechizo.

2. Modelar las siguientes funciones:
    a. **poder**
    El poder de un mago es su salud sumada al resultado de multiplicar su edad por la cantidad de hechizos que conoce.

    b. **daño**
    Esta función retorna la cantidad de vida que un mago pierde si le lanzan un hechizo.

    c. **diferenciaDePoder**
    La diferencia de poder entre dos magos es el valor absoluto de la resta del poder de cada uno. Esto siempre retorna un número positivo.

3. Dada una Academia, la cual representamos con el siguiente tipo de dato:

```
data Academia = Academia {
  magos :: [Mago],
  examenDeIngreso :: Mago -> Bool
}
```

Se pide escribir el código necesario para realizar las siguientes **consultas** sin usar recursividad:

a. Saber si hay algún mago sin hechizos cuyo nombre sea "Rincenwind".
b. Saber si todos los magos viejos (>50) son ñoños. Esto ocurre si tienen más hechizos que salud.
c. Conocer la cantidad de magos que no pasarían el examen de ingreso si lo volvieran a rendir.
d. Averiguar la sumatoria de la edad de los magos que saben más de 10 hechizos.

4. Dadas las siguientes funciones:

```
maximoSegun criterio valor comparables =
      foldl1 (mayorSegun $ criterio valor) comparables

mayorSegun evaluador comparable1 comparable2
      | evaluador comparable1 >= evaluador comparable2 = comparable1
      | otherwise                                      = comparable2
```

Usar la función `maximoSegun` para definir las siguientes funciones, sin repetir código ni definir funciones auxiliares:

i. **mejorHechizoContra**
Dados dos magos, retorna el hechizo del segundo que le haga más daño al primero.

ii. **mejorOponente**
Dado un mago y una academia, retorna el mago de la academia que tenga la mayor diferencia de poder con el mago recibido.

5. Definir la siguiente función sin utilizar recursividad:
noPuedeGanarle
Decimos que el segundo mago no puede ganarle al primero si, luego de hechizarlo con todos los hechizos que conoce (uno atrás del otro) la salud del primer mago sigue siendo la misma.