First Fantasy

La empresa Cuadrado Enix se encuentra al borde de la quiebra y nos contrató para crear un juego de fantasía con un fuerte énfasis en su historia. Este va a ser un juego de rol y, como esperaríamos, los personajes van a poder interactuar de diferentes formas con el mundo.

Los personajes tienen atributos esenciales: vida, fuerza y resistencia, y una serie de objetos que utilizan para atacar.

- La vida se mide en puntos y nos indica qué tan saludable está un personaje en un momento dado.
- La resistencia se utiliza para calcular la **defensa** del personaje, la cual es la resistencia más la vida del personaje.
- La fuerza nos indica el poderío físico de un personaje. El **ataque** se calcula como la fuerza que tiene el personaje modificado por todos los los poderes de los objetos que el personaje tenga, aplicados en orden.

Algunos objetos que existen en el mundo son:

```
espadaOxidada = (1.2*)
katanaFilosa  = (10+).(0.9*)
dagaLambdica cm = ((1+cm)/100*)
anilloParadigmatico = sqrt
baculoDuplicador x = x * 2
espadaMaldita = espadaOxidada.dagaLambdica 89
```

Y algunos personajes pueden ser:
Aeris, con vida 10000, 2300 de resistencia y 1100 de fuerza, tiene un baculoDuplicador.
Sephiroth, con vida 2500, 2000 de resistencia y fuerza 1000, tiene una espadaOxidada, una katanaFilosa y una dagaLambdica de 50 cm.

Se pide:

1. Calcular el ataque y la defensa de un personaje.
2. Modelar que un personaje ataque a otro. El daño que el atacante hace sobre el atacado es igual al ataque del atacante menos la defensa del atacado (no puede ser negativo), y debe restarse de la vida de quién se defiende.
3. Saber si a un personaje le conviene pelear contra otro. A un personaje le conviene pelear contra otro si su poder de ataque al robarle todos los objetos al contrario es mayor que su poder de ataque original.
4. Modelar que un personaje pelee contra otro. Empieza atacando el primero, y ataca una vez cada uno, hasta que uno de ellos pierda (su vida llegue a cero). Se retorna

al ganador, cuya vida quedará reducida debido a los daños recibidos, pero a cambio tomará los objetos del perdedor.

5. En este tipo de juegos suele haber aprendices y maestros artesanos. Éstos se dedican a hacer mejores objetos para los aventureros. Hay muchos tipos, pero algunos de los más usuales son:
    - Los **aprendices**, que combinan todos los objetos del personaje en uno nuevo cuyo efecto es la acumulación de todos los efectos.
    - Los **maestros artesanos**, que hacen lo mismo que los aprendices, pero además aplican al nuevo objeto un 10% extra de ataque tantas veces como años de experiencia tenga haciendo artesanías
    - Los **estafadores**, que reemplazan todos los objetos del usuario por un objeto que no hace nada
    - Inventar un nuevo tipo de artesano que, similar a los anteriores, haga algo con los objetos del personaje. Debe utilizar listas por comprensión de una forma que no sea trivial.

    Saber de una lista de artesanos:
    1. Todos aquellos que hacen que el nuevo ataque del personaje sea superior a un valor dado.
    2. Si todos le convienen al personaje.

6. Si existe un personaje con una cantidad infinita de objetos, ¿de qué artesanos puede solicitar los servicios? Mostrar un caso en donde finalice correctamente y otro donde no finalice, justificando por qué. Indicar el nombre del concepto que hace posible esto y explicarlo brevemente.

7. Para pensar: A fin de que un personaje pueda obtener un objeto nuevo y curarse 100 de vida, se propone la siguiente solución:
    agregarObjeto obj (Personaje a b c) = Personaje ((+ 100).a) b (\x-> obj x):c
    ¿Funciona correctamente? ¿Es expresiva? ¿Qué conceptos del paradigma utiliza? Justificar todas las respuestas y proponer una solución mejor en caso de considerarlo necesario.

Probar los diferentes puntos, agregando (o modificando) los personajes

Posible solucion

```
import Text.Show.Functions

data Personaje = UnPersonaje { vida :: Float,
                               resistencia :: Float,
                               fuerza :: Float,
                               objetos :: [(Float -> Float)]} deriving (Show)

aeris = UnPersonaje 500 500 2100 [baculoDuplicador]
sephiroth = UnPersonaje 2000 1000 2500 [espadaOxidada, katanaFilosa, dagaLambdica
50]

espadaOxidada = (1.2*)
katanaFilosa = (10+) . (0.9*)
dagaLambdica cm = ((1+cm)/100*)
anilloParadigmatico = sqrt
baculoDuplicador  x = x * 2
espadaMaldita = espadaOxidada.dagaLambdica 89
espadaLegendaria = (1*)

-- Punto 1

defensa personaje = resistencia personaje + vida personaje

ataque personaje = foldl (\x f -> f x) (fuerza personaje) (objetos personaje)
--variantes
--ataque (UnPersonaje _ _ fuerza objetos) = (head (combinar objetos)) fuerza
--ataque personaje = foldr ($) (fuerza personaje) (objetos personaje)

{-
*Main> ataque sephiroth
1389.0
*Main> defensa sephiroth
3000.0
-}
-- Punto 2

atacar personaje1 personaje2 = personaje2 {vida = (vida personaje2) - max 0 (ataque
personaje1 - defensa personaje2)}

{-
*Main> atacar aeris sephiroth
UnPersonaje {vida = 800.0, resistencia = 1000.0, fuerza = 2500.0, objetos =
[<function>,<function>,<function>]}
-}
```

```
-- Punto 3

convienePelear personaje1 personaje2 = ataque personaje1 < ataque (robarObjetos
personaje1 personaje2)

robarObjetos personaje1 personaje2 = personaje1 {objetos = objetos personaje1 ++ objetos
personaje2}


{-
*Main> convienePelear aeris sephiroth
False
-}

-- Punto 4

pelear personaje1 personaje2
--Agregado para evitar que peleen los sin vida
-- | vida personaje1 <= 0 = error "Sin vida no se puede empezar una pelea"
-- | vida personaje2 <= 0 = error "Sin vida no se puede ser atacado en una pelea"
 | vida (atacar personaje1 personaje2) <= 0 = robarObjetos personaje1 personaje2
 | otherwise = pelear (atacar personaje1 personaje2) personaje1

{-
--Variante sutil
pelear personaje1 personaje2
 | vida atacado <= 0 = robarObjetos personaje1 personaje2
 | otherwise = pelear atacado personaje1
   where atacado = atacar personaje1 personaje2
-}


{-
*Main> pelear aeris sephiroth
UnPersonaje {vida = 117.900024, resistencia = 500.0, fuerza = 2100.0, objetos =
[<function>,<function>,<function>,<function>]}
-}

-- Punto 5

aprendiz personaje = personaje {objetos = combinar (objetos personaje)}
combinar objetos = [foldl1 (.) objetos]

maestroArtesano experiencia personaje = aprendiz personaje{ objetos = potenciar
experiencia (objetos personaje)}
potenciar veces objetos = objetos ++ replicate veces (1.1*)
```

```
estafador personaje = personaje {objetos = [id]}

--Variante con recursividad
--maestroArtesano experiencia personaje = mejorar experiencia (aprendiz personaje)
--mejorar 0 personaje = personaje
--mejorar experiencia personaje = mejorar (experiencia - 1) personaje{objetos =
((1.1*):objetos personaje)}

--Otra variante
--aprendiz = artesano combinar
--maestroArtesano experiencia = artesano (potenciar experiencia).aprendiz
--estafador = artesano (\_ -> [id])
--artesano accion personaje = personaje {objetos = accion (objetos personaje)}

--Otra mas
--estiloAprendiz = combinar
--estiloMaestroArtesano experiencia = potenciar experiencia.combinar
--estiloEstafador = (\_ -> [id])
--artesano estilo personaje = personaje {objetos = estilo (objetos personaje)}

artesanosMayores nuevoAtaque artesanos personaje = filter (mayorAtaque personaje
nuevoAtaque) artesanos

mayorAtaque personaje valor artesano = ataque (artesano personaje) > valor

artesanosConvienen artesanos personaje = all (mayorAtaque personaje (ataque personaje))
artesanos

{-
*Main> ataque (maestroArtesano 10 aeris)
10893.721
*Main> ataque (aprendiz aeris)
4200.0
*Main> ataque (estafador aeris)
2100.0
*Main> ataque (maestroArtesano 10 sephiroth)
3583.5842
*Main> ataque (aprendiz sephiroth)
1389.0
*Main> ataque (estafador sephiroth)
2500.0

*Main> artesanosConvienen [aprendiz, estafador, maestroArtesano 20] sephiroth
False
-}
```