Aprenda a jugar golf con Lee Carvallo

Lisa Simpson se propuso desarrollar un programa que le permita ayudar a su hermano a vencer a su vecino Todd en un torneo de minigolf. Para hacerlo más interesante, los padres de los niños hicieron una apuesta: **el padre del niño que no gane** deberá cortar el césped del otro usando un vestido de su esposa.

De los participantes nos interesará el nombre del jugador, el de su padre y sus habilidades (fuerza y precisión).

```
-- Modelo inicial
data Jugador = UnJugador {
  nombre :: String,
  padre :: String,
  habilidad :: Habilidad
} deriving (Eq, Show)

data Habilidad = Habilidad {
  fuerzaJugador :: Int,
  precisionJugador :: Int
} deriving (Eq, Show)

-- Jugadores de ejemplo
bart = UnJugador "Bart" "Homero" (Habilidad 25 60)
todd = UnJugador "Todd" "Ned" (Habilidad 15 80)
rafa = UnJugador "Rafa" "Gorgory" (Habilidad 10 1)

data Tiro = UnTiro {
  velocidad :: Int,
  precision :: Int,
  altura :: Int
} deriving (Eq, Show)

type Puntos = Int

-- Funciones útiles
between n m x = elem x [n .. m]

maximoSegun f = foldl1 (mayorSegun f)
mayorSegun f a b
  | f a > f b = a
  | otherwise = b
```

También necesitaremos modelar los palos de golf que pueden usarse y los obstáculos que deben enfrentar para ganar el juego.

1. Sabemos que cada palo genera un efecto diferente, por lo tanto elegir el palo correcto puede ser la diferencia entre ganar o perder el torneo.
    a. Modelar los palos usados en el juego que a partir de una determinada habilidad generan un **tiro** que se compone por velocidad, precisión y altura.
        i. El **putter** genera un tiro con velocidad igual a 10, el doble de la precisión recibida y altura 0.

   ii. La **madera** genera uno de velocidad igual a 100, altura igual a 5 y la mitad de la precisión.
   iii. Los **hierros**, que varían del 1 al 10 (número al que denominaremos n), generan un tiro de velocidad igual a la fuerza multiplicada por n, la precisión dividida por n y una altura de n-3 (con mínimo 0). Modelarlos de la forma más genérica posible.
b. Definir una constante **palos** que sea una lista con todos los palos que se pueden usar en el juego.

2. Definir la función **golpe** que dados una persona y un palo, obtiene el tiro resultante de usar ese palo con las habilidades de la persona.
   Por ejemplo si Bart usa un putter, se genera un tiro de velocidad = 10, precisión = 120 y altura = 0.
3. Lo que nos interesa de los distintos obstáculos es si un tiro puede superarlo, y en el caso de poder superarlo, cómo se ve afectado dicho tiro por el obstáculo. En principio necesitamos representar los siguientes obstáculos:
   a. Un túnel con rampita sólo es superado si la precisión es mayor a 90 yendo al ras del suelo, independientemente de la velocidad del tiro. Al salir del túnel la velocidad del tiro se duplica, la precisión pasa a ser 100 y la altura 0.
   b. Una laguna es superada si la velocidad del tiro es mayor a 80 y tiene una altura de entre 1 y 5 metros. Luego de superar una laguna el tiro llega con la misma velocidad y precisión, pero una altura equivalente a la altura original dividida por el **largo de la laguna**.
   c. Un hoyo se supera si la velocidad del tiro está entre 5 y 20 m/s yendo al ras del suelo con una precisión mayor a 95. Al superar el hoyo, el tiro se detiene, quedando con todos sus componentes en 0.

   Se desea saber cómo queda un tiro luego de intentar superar un obstáculo, teniendo en cuenta que en caso de no superarlo, se detiene, quedando con todos sus componentes en 0.
4.
   a. Definir **palosUtiles** que dada una persona y un obstáculo, permita determinar qué palos le sirven para superarlo.
   b. Saber, a partir de un conjunto de obstáculos y un tiro, cuántos obstáculos consecutivos se pueden superar.
      Por ejemplo, para un tiro de velocidad = 10, precisión = 95 y altura = 0, y una lista con dos túneles con rampita seguidos de un hoyo, el resultado sería 2 ya que la velocidad al salir del segundo túnel es de 40, por ende no supera el hoyo.
      BONUS: resolver este problema sin recursividad, teniendo en cuenta que existe una función `takeWhile :: (a -> Bool) -> [a] -> [a]` que podría ser de utilidad.
   c. Definir **paloMasUtil** que recibe una persona y una lista de obstáculos y determina cuál es el palo que le permite superar más obstáculos con un solo tiro.
5. Dada una lista de tipo `[(Jugador, Puntos)]` que tiene la información de cuántos puntos ganó cada niño al finalizar el torneo, se pide retornar la lista de padres que pierden la apuesta por ser el “padre del niño que no ganó”. Se dice que un niño ganó el torneo si tiene más puntos que los otros niños.

Lee Carvalo: La pelota está en el estacionamiento.
Quieres jugar de nuevo?

Bip

Lee Carvalo: Escogiste... No.