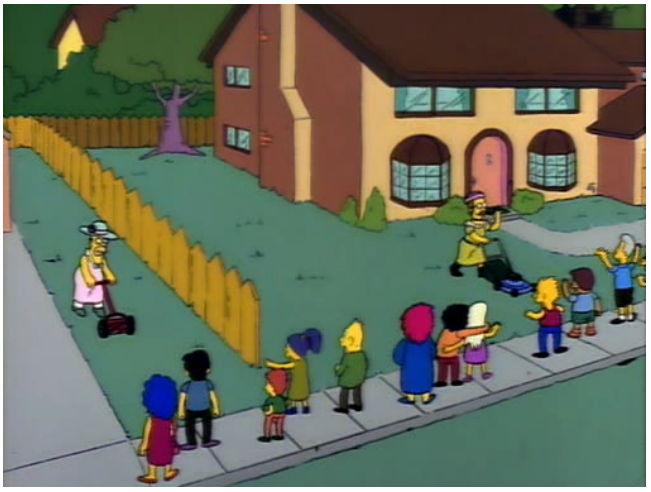
Aprenda a jugar golf con Lee Carvallo



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 mujer.
De los participantes nos interesará el nombre del jugador, el de su padre y sus habilidades (fuerza y precisión). También se modelan los palos de golf que pueden usarse y los obstáculos que deben enfrentar para ganar el juego.

```
bart = ("Bart","Homero",(25,60))
todd = ("Todd","Ned",(15,80))
rafa = ("Rafa","Gorgory",(10,1))
nombre (n,_,_) = n
padre (_,p,_) = p
habilidad (_,_,h) = h
```

Se sabe que un obstáculo es una tupla de tipo **`( Tiro -> Bool , Tiro -> Tiro )`**
El primer elemento indica la condición a aplicar sobre un tiro (tupla de velocidad, precisión y altura) para determinar si el obstáculo puede ser superado y el segundo es el efecto a aplicar sobre el tiro cuando el obstáculo sea superado.

```
laguna largo = ((\(v,_,a)-> v>80 && between 10 50 a),(\(v,p,a) -> (v,p,a `div` largo)))
tunelConRampita = ((\(_,p,a) -> p>90 && a==0), (\(v,_,a) -> (v*2,100,0)) )
hoyo = ((\(v,p,a) -> between 5 20 v && p>95 && a==0), (\ _ -> (0,0,0)))
between n m x = elem x [n .. m]

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

Se pide resolver la siguiente funcionalidad aplicando los conceptos del paradigma funcional y sin repetir código. La solución debe demostrar buen dominio de **composición, orden superior y aplicación parcial** así como el uso de buenas abstracciones, **no se puede usar recursividad salvo en el punto 3a**.

1. Sabemos que cada palo genera un efecto diferente, elegir el palo correcto puede ser la diferencia entre ganar o perder el torneo.
   Definir las funciones para modelar los palos usados en el juego que dada una tupla habilidad retorna un tiro que se compone por velocidad, precisión y altura.
   - El **putter** genera un tiro con velocidad igual a 10, el doble de la precisión recibida y altura 0
   - La **madera** genera uno de velocidad igual a 100, altura igual a 5 y la mitad de la precisión
   - 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 al cuadrado. Modelarlos con una sola función hierro.

   Para los siguientes puntos se podrá disponer de la función palos definida como:
   `palos = putter : madera : map hierro [1 .. 10]`
   La lista palos puede usarse en la definición de otras funciones, son todos los palos disponibles del juego.
2.
   a. Definir la función **golpe** que dados una persona y un palo, obtiene el tiro resultante de usar el palo

con las habilidades de la persona.

```
> golpe bart putter
(10,120,0)
```

b. Saber si un tiro puede superar un obstáculo; esto sucede cuando el mismo cumple la condición del obstáculo.

```
> puedeSuperar hoyo (7,100,0)
True
```

c. Definir **palosUtiles** que dada una persona y un obstáculo, permita determinar qué palos le sirven para superarlo.

```
> palosUtiles bart hoyo
[putter]
```
[1]

d. Dada una lista de persona y una lista de obstáculos, obtener los nombres de las personas que pueden superar todos los obstáculos, lo cual sucede si para cada obstáculo hay al menos un palo útil para superarlo.

```
> nombresDeLosQuePuedenSuperarTodos [tunelConRampita, laguna 3, hoyo]
      [bart,rafa]
["Bart"]
```

3.

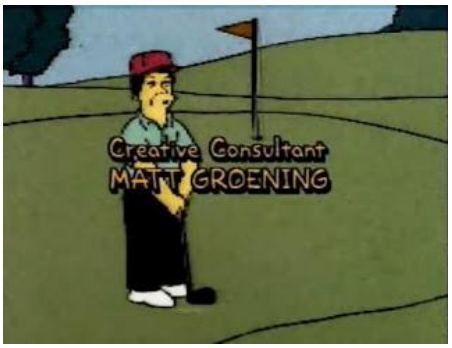
a. Saber cuántos obstáculos seguidos puede superar un tiro

```
> cuantosObstaculosSupera (10,120,0) [tunelConRampita, tunelConRampita, hoyo]
2 → la velocidad al salir del segundo túnel es de 40, no entra en el hoyo
```

b. 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.

4. Se tiene una función `puntosGanados :: [Obstáculo] -> Persona -> Int` que devuelve la cantidad de puntos ganados por la persona al pasar por los obstáculos.
Dada una lista de niños y otra de obstáculos, retornar la lista de padres que pierden la apuesta por ser el padre del niño que no ganó. Se dice que un niño gana si puede superar todos los obstáculos y tiene más puntos que los otros niños.

```
> pierdenLaApuesta [bart, todd] [tunelConRampita, laguna 1, hoyo]
["Homero", "Ned"] → asumiendo que terminan con el mismo puntaje, bart y todd
empatan (ninguno gana)
> pierdenLaApuesta [bart, rafa] [tunelConRampita, hoyo]
["Gorgory"] → rafa no puede pasar todos los obstáculos, gana bart
```



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

Bip

Lee Carvalo: Escogiste... No.

[1] El palo no puede imprimirse realmente por ser una función, el ejemplo es a modo ilustrativo