Carrera de Chocobos

Las carreras de chocobos son un entretenimiento cada día más popular, y por lo tanto ya es hora de armar un programa que nos ayude a analizarlas como es debido. Elegimos hacerlo en Haskell, básicamente por inercia (y... ya que lo venimos usando hace 2 meses, sigamos con eso).

Las pistas por las que nuestros emplumados amigos deben correr van a estar representadas por listas de tramos, cada tramo a su vez será representado por una tupla (distancia, correcciónDeVelocidad).

```
bosqueTenebroso =
        [(100, f1), (50, f2), (120, f2), (200, f1), (80, f3)]
pantanoDelDestino =
        [(40, f2), (90, (\(f,p,v)-> f + p + v)), (120, fuerza), (20, fuerza)]

f1 chocobo = velocidad chocobo * 2
f2 chocobo = velocidad chocobo + fuerza chocobo
f3 chocobo = velocidad chocobo / peso chocobo
```

Tenemos los chocobos (esenciales para una carrera de chocobos): el amarillo, el negro, el blanco y el rojo. Cada uno tiene distintas características, modeladas por medio de una tupla (fuerza, peso, velocidad).

```
amarillo = (5, 3, 3)
negro    = (4, 4, 4)
blanco   = (2, 3, 6)
rojo     = (3, 3, 4)
```

Así mismo, están las funciones de acceso a la tupla:

```
fuerza     (f,_,_) = f
peso       (_,p,_) = p
velocidad  (_,_,v) = v
```

Finalmente, estos chocobos están dirigidos por los 4 jinetes:

```
apocalipsis = [("Leo", amarillo), ("Gise", blanco), ("Mati", negro), ("Alf",rojo)]
```

Disponemos de esta función a modo de ayuda que, a partir de una lista y un criterio de ordenamiento, nos devuelve la versión equivalente a esa lista pero con los elementos ordenados por el criterio dado.

```
quickSort _ [] = []
quickSort criterio (x:xs) =
        (quickSort criterio . filter (not . criterio x)) xs
        ++ [x] ++
        (quickSort criterio . filter (criterio x)) xs
```

Ejemplo de uso:

```
> quickSort (>) [3,8,7,20,2,1]
[20,8,7,3,2,1]
```

Notas:

Deberán utilizar correctamente al menos una vez cada uno de las siguientes conceptos:

- **Orden superior**
- **Listas por comprensión**
- **Composición**
- **Aplicación parcial**

No se pueden definir funciones recursivas en más de un punto de los desarrollados.

Se pide desarrollar las siguientes funciones:

1. Definir dos funciones `mayorSegun` y `menorSegun` que, dados una función y dos valores, nos dice si el resultado de evaluar la función para el primer valor es mayor / menor que el resultado de evaluar la función para el segundo.
```
> mayorSegun length bosqueTenebroso pantanoDelDestino
True          (tiene 5 tramos el bosque y 4 tramos el pantano)
```

2.
   a. Saber el tiempo que tarda un chocobo en recorrer un tramo. El mismo está dado por la distancia del tramo dividido por la velocidad corregida para el chocobo.
```
> tiempo amarillo (head bosqueTenebroso)
16
```

   b. Determinar el tiempo total de un chocobo en una carrera.
```
> tiempoTotal bosqueTenebroso amarillo
150
```

3. Obtener el podio de una carrera, representado por una lista ordenada de los 3 primeros puestos de la misma, en base a una lista de jinetes y una pista. El puesto está dado por el tiempo total, de menor a mayor y se espera obtener una lista de jinetes.
```
> podio bosqueTenebroso apocalipsis
[("Gise",(2,3,6)),("Mati",(4,4,4)),("Alf",(3,3,4))]         (ver también ejemplo del punto 6)
```

4.
   a. Realizar una función que dado un tramo y una lista de jinetes, retorna el nombre de aquel que lo recorrió en el menor tiempo.
```
> elMejorDelTramo (head bosqueTenebroso) apocalipsis
"Gise"          (Gise tarda 8, mientras que Leo tarda 16 y Mati y Alf tardan 12)
```

   b. Dada una pista y una lista de jinetes, saber el nombre del jinete que ganó más tramos (que no quiere decir que haya ganado la carrera).
```
> elMasWinner pantanoDelDestino apocalipsis
"Leo"           (gana 2 tramos, el resto gana 1 o ninguno)
```

5. Saber los nombres de los jinetes que pueden hacer un tramo dado en un tiempo indicado máximo..
```
> quienesPueden (head bosqueTenebroso) 12 apocalipsis
["Gise","Mati","Alf"]         (ver 4.a)
```

6. Obtener las estadísticas de una carrera, dada la pista y la lista de jinetes. Estas estadísticas deben estar representadas por una lista de tuplas, cada tupla siendo de la forma: (nombre, tramosGanados, tiempoTotal)
```
> estadisticas bosqueTenebroso apocalipsis
[("Leo",0,150),("Gise",3,85),("Mati",2,138),("Alf",0,141)]
```

7. Saber si una carrera fue pareja. Esto es así si cada chocobo tuvo un tiempo total de hasta 10% menor que el que llegó a continuación.
```
> fuePareja bosqueTenebroso apocalipsis
False           (entre Gise y Mati, 1ª y 2° respectivamente, hay más de 10% de diferencia)
```

8. Definir un chocobo plateado que tenga las mejores características de los otros (mayor fuerza, menor peso, mayor velocidad), teniendo en cuenta **que no sea necesario cambiar su definición si se altera un valor de los anteriores**.
```
> plateado
(5,3,6)
```

9. Definir el tipo de funcionHeavy:
```
funcionHeavy x y z
        | (fst . head) x < snd y = map z x
        | otherwise = filter (fst y ==) (map z x)
```