Eliminatorias

Se está organizando un torneo en el cual muchos participantes compiten en diversos desafíos, volviéndose más experimentados luego de cada desafío ganado.

De cada **participante** nos interesará su **nombre**, la cantidad de **experiencia** que tiene, su nivel de **inteligencia** y **destreza física**, y el **rol** que desempeña. Un participante podría ser más apto para un rol que para otro, con lo cual será un factor importante elegir el rol más adecuado para encarar cada desafío.

Algunos roles que necesitamos representar inicialmente son los siguientes:

- Indeterminado: la aptitud de un participante para este rol equivale a la suma entre su inteligencia y su destreza física.
- Soporte: la aptitud de un participante para este rol equivale a su inteligencia multiplicada por 7 sumada a su experiencia.
- Primera línea: para este rol se indicará también la potencia del arma que deberá manejar el participante; la aptitud de un participante para este rol equivale a la suma de su destreza física y la potencia del arma, todo eso multiplicado por la experiencia del participante dividida por 100.

Resolver los siguientes requerimientos de modo que se aproveche todo lo posible del uso de **composición, aplicación parcial y orden superior**. **Explicitar los tipos** de todas las funciones desarrolladas. **Evitar repetir lógica y soluciones poco declarativas**.

1. A partir de la información del dominio previamente indicada:
    a. Modelar a los participantes y los roles, incluyendo los distintos roles explicados, de modo que sea posible incorporar nuevos roles fácilmente.
    b. Declarar una constante de tipo Participante que tenga 10 puntos de experiencia, inteligencia igual a 20, destreza física igual a 12 y el rol ***indeterminado***.
    c. Desarrollar una función para conocer el **poder** de un participante, que se calcula como su experiencia multiplicada por la aptitud que tiene ese participante para el rol que desempeña. *Ejemplo: para el participante del punto anterior, su poder debería ser 10 * (20 + 12).*
    d. Inventar un nuevo rol que tenga lógica distinta a la de los roles existentes para determinar la aptitud de un participante.
    ¿Tuviste que hacer algún cambio sobre lo desarrollado en los puntos anteriores para conseguirlo? Explicá brevemente por qué sí o por qué no fue necesario.

Con este modelo en mente, y sabiendo que a su vez disponemos de las siguientes funciones para usar:

```
maximoSegun :: Ord a => (b -> a) -> [b] -> b
maximoSegun f = foldl1 (mayorSegun f)

mayorSegun :: Ord a => (p -> a) -> p -> p -> p
mayorSegun f a b
  | f a > f b = a
  | otherwise = b
```

2. Desarrollar una función para que un participante elija un nuevo rol a partir de un conjunto de roles disponibles, cambiando su rol actual por el elegido. El rol a elegir para el participante debería ser aquel para el cual sea más apto.

Mostrar una **consulta** que permita conocer el poder del participante del punto 1b luego de elegir un rol, incluyendo entre los roles disponibles los desarrollados en el punto 1a y el punto 1d.

3. A continuación queremos prepararnos para resolver los desafíos que incorporaremos más adelante. Para ello se pide desarrollar funciones para...
    a. Saber si un participante se encuentra entre un grupo de participantes. Debería cumplirse si entre esos participantes hay alguno que tenga el mismo nombre que el que nos interesa.

    b. Calcular la **experiencia a ganar** luego de completar un desafío. Para ello se proveerá una lista con todos los participantes de un desafío y una lista con aquellos que consideramos ganadores.
    La experiencia a ganar será el resultado de sumar la experiencia de aquellos participantes que no se encuentren entre los ganadores, dividido[1] por la cantidad de ganadores, más 100 puntos extra.
    Ejemplo: si tenemos 5 participantes cuyos niveles de experiencia son: 1000, 1500, 1200, 150 y 1800, y los primeros dos no se encuentran entre los ganadores, la experiencia a ganar debería ser: (1000 + 1500) / 3 + 100.

    c. **Repartirles la experiencia** a los ganadores. Nuevamente, a partir de una lista con todos los participantes de un desafío y una lista con aquellos que consideramos ganadores, queremos hacer que los ganadores ganen la experiencia que corresponda en base al cálculo del punto anterior.

4. Necesitamos desarrollar lo necesario para que un conjunto de participantes encaren un desafío. Para ello incorporaremos a los desafíos a nuestro modelo, de esta forma:

```
data Desafio = Desafio {
      rolesDisponibles :: [Rol],
      pruebaASuperar :: Participante -> Bool }
```

    a. Resolver la lógica para que un conjunto de participantes encaren un desafío, donde lo esperado es que primero **elijan el rol para el cual sean más aptos** a partir de aquellos que están disponibles para el desafío, luego **traten de superar la prueba** de dicho desafío y finalmente se **reparta la experiencia que corresponda** entre los ganadores (los que pasaron la prueba).
    Nota: se espera que el rol que tengan los ganadores luego del desafío sea el elegido para el mismo, no el que tenían antes de encarar el desafío.

    b. Usando lo desarrollado en el punto anterior mostrar qué consultas pueden hacerse para:
        i. Saber si alguno de los ganadores de un desafío tiene más de 1000 puntos de experiencia.
        ii. Saber quién es el más poderoso de aquellos que hayan ganado un desafío.
    Nota: No importa cuáles sean concretamente el desafío y los participantes, sólo el uso.

    c. Para cada una de las consultas del punto anterior explicar si podrían completarse si el conjunto de participantes que encara el desafío fuese infinito. Sé puntual en tus respuestas respecto al motivo por el cual podrían o no terminar.

5. Finalmente necesitamos hacer que un conjunto de participantes compitan en un torneo. Sabemos que un torneo se compone por muchos desafíos, a partir de los cuales esos participantes irán quedando eliminados progresivamente. Necesitamos conocer cómo quedarán aquellos participantes que logren terminar el torneo, luego de encarar los desafíos uno tras otro.

[1] Vale usar división entera, para evitar problemas con tipos numéricos irrelevantes.