Karrrrs!

Índice

1 Cambios al dominio

2 Temas a evaluar

3 Entrega 1
- 3.1 Punto 1: Modelado de carreras
- 3.1 Punto 2: Modelado de trampas
- 3.3 Punto 3: ¡A correr!
- 3.4 Punto 4: ¿Quién gana?
- 3.5 Punto 5: elGranTruco
- 3.6 Punto 6: Una gran carrera

4 Casos de prueba

5 Test unitarios automatizados

1 Cambios al dominio

Como se imaginarán, los autos y sus trucos no nos sirven de mucho si no podemos hacer que los autos los utilicen en carreras.

De las carreras conocemos la cantidad de vueltas que se deben correr, la longitud de la pista en kilómetros, los nombres de los integrantes del público, una trampa y los participantes, claro.

Teniendo en cuenta esto, se pide realizar cambios a:
deReversa: Finalmente deReversa no tomará en cuenta la longitud de la pista, si no que aumentará su nafta en una quinta parte de la velocidad.

2 Temas a evaluar

- Orden superior
- Modelado de información
- Refactor | Modificaciones a un código existente
- Recursividad
- Evaluación diferida
- Listas

3 Entrega 1

3.0 Punto 0:

1) Modificar el truco **deReversa** de acuerdo a lo indicado.

3.1 Punto 1: Modelado de carreras

1) Modelar el tipo de dato carrera.
2) Modelar la carrera **potreroFunes** en la que el largo de la pista es de 5,0 kilómetros, se deben dar 3 vueltas, en el público están Ronco, Tinch, y Dodain, y en la que participan **RochaMcQueen**, **Biankerr**, **Gushtav** y **Rodra** (quienes salen de la línea de meta en ese mismo orden). Su trampa es sacarAlPistero.

3.2 Punto 2: Modelado de trampas

Como dijimos antes, cada carrera tiene una trampa específica. Algunas de las trampas son:

- **sacarAlPistero**: el primer participante queda fuera de la competencia.
- **lluvia:** baja 10 km/h de velocidad a todos los participantes
- **neutralizarTrucos**: inutiliza a todos los participantes. En otras palabras, todos tendrán el truco **inutilidad**, que no hace nada.
- **pocaReserva**: los participantes con menos de 30 litros de nafta quedan fuera.
- **podio:** solamente deja seguir compitiendo a los primeros 3 participantes de la carrera.

3.3 Punto 3: ¡A correr!

Cuando los participantes corren una carrera, al transcurrir cada vuelta, suceden tres cosas:

1) Se restará del combustible de cada auto según el siguiente cálculo: 1 litro de nafta por cada km/h de velocidad del auto por cada 10 kilómetros de la pista. Es decir, si la pista tiene 5 kilómetros y voy a 100 Km/hora, el consumo es de 50 litros (5 / 10 * 100) (la fórmula sería kilómetros de la pista / 10 * velocidad).
2) Los participantes cuyo enamorade esté en el público realizan su truco (en caso de poder).
3) Todos los participantes sufren la trampa de la carrera.

Se pide:

1) Implementar la función *darVuelta* que dada una carrera, nos responda como queda la misma luego de dar una vuelta.
2) Implementar la función correrCarrera que dada una carrera, nos devuelva a la carrera luego de dar todas sus vueltas.[1]

3.4 Punto 4: ¿Quién gana?

Como es de esperarse, si corremos una carrera, alguien la va a ganar, por esto te pedimos definir la función *quienGana* que dada una carrera, nos devuelva al auto ganador. Consideramos que gana el auto que tiene la mayor velocidad al terminar la carrera.

3.5 Punto 5: elGranTruco

Llegó el momento de conocer al truco más importante de todos (truco, no Trucco): modelar **elGranTruco**, el cual recibe una lista de trucos y aplica todos estos a un auto.

[1] Para esto les van a servir las funciones **iterate** y **!!**

```
λ> elGranTruco [nitro, deReversa, impresionar] rodra
Auto {nombre = "rodra", nivelDeNafta = 13, velocidad = 130,
enamorade = "Taisa", truco = <function>}
```

3.6 Punto 6: Una gran carrera

Luego tenemos la carrera ultra suprema de las altas ligas que tiene una cantidad infinita de participantes!
De ella queremos saber:
a) ¿Podemos correrla?
b) ¿Podemos conocer el primer participante luego de 2 vueltas?
c) ¿Podemos dar la primera vuelta de la carrera?

4 Casos de prueba

CASOS DE PRUEBA	
PUNTO 3.0	
Descripción	**Resultado Esperado**
Consultar la nafta de rochaMcQueen luego de hacer su truco favorito (nivelDeNafta . truco rochaMcQueen) rochaMcQueen	300 (nafta inicial más un quinto de su velocidad)
Consultar la nafta de rodra tras impresionar	10
PUNTO 3.2	
Descripción	**Resultado Esperado**
Consultar la cantidad de participantes luego de sacarAlPistero en potreroFunes	3
rochaMcQueen ya no participa en potreroFunes tras sacarAlPistero	False
Consultar la cantidad de participantes luego de pocaReserva en potreroFunes	3
El participante llamado “rodra” ya no debería estar entre los participantes de potreroFunes luego de aplicar pocaReserva	False
Consultar la cantidad de participantes luego de aplicar podio en potreroFunes	3
Consultar la velocidad del último participante de potreroFunes (rodra) luego de la lluvia	40

PUNTO 3.3	
Descripción	**Resultado Esperado**
Consultar el nivel de nafta del primer participante (biankerr porque rochaMcQueen quedó afuera) luego de dar una vuelta en potreroFunes.	490
Consultar la velocidad del primer participante (biankerr porque rochaMcQueen quedó afuera) luego de dar una vuelta en potreroFunes.	40 (porque aplica su truco)
Consultar la cantidad de participantes tras dar dos vueltas en potrero funes	2
Luego de 2 vueltas en potreroFunes, consultar el nivelDeNafta del primer participante (gushtav)	*70*
Rodra debe ser el único participante luego de correr la carrera de potreroFunes	
PUNTO 3.4	
Descripción	**Resultado Esperado**
Consultar el ganador de potreroFunes	*rodra*
PUNTO 3.5	
Descripción	**Resultado Esperado**
Consultar la velocidad de rodra tras realizar elGranTruco con nitro, deReversa e impresionar	*130*
Consultar el nivelDeNafta de rodra tras realizar elGranTruco con nitro, deReversa e impresionar	*13*

5 Test unitarios automatizados

En esta entrega vamos a pedir que realicen tests automatizados. Para ello, el archivo .hs se debe llamar Kars.hs, y la primera línea debe escribirse de la siguiente manera:

```
module Kars where
```

(luego escriben todas las funciones propias del TP)

Luego, hay que crear un archivo para los tests que se debe llamar KarsTest.hs, y en las primeras líneas deben escribir:

```
import Kars
import Test.Hspec
```

Vamos a necesitar que descarguen Haskell Platform, que incluye Cabal, una herramienta de manejo de dependencias de Haskell, y que instalen Hspec, con los que vamos a correr las pruebas unitarias automatizadas de tu TP.
Para instalar Hspec los pasos son:

1. Abrir la consola del sistema operativo.
2. Ejecutar el comando: **`cabal update && cabal install hspec.`**

Luego, para correr los tests que hicieron deben:

1. Abrir la consola del sistema operativo en la carpeta que tienen el TP.
2. Ejecutar el comando: **`runhaskell KarsTest.hs`**