Tema B

Ejercicio 1:

Una empresa de peajes está desarrollando un sistema de gestión para las cabinas en Haskell y como parte de ese trabajo necesita representar los rodados que pasan por la ruta. Los móviles reconocidos hasta el momento son: `Automovil, Moto, Bus, Camion`. Cada uno tiene sus características especiales a saber:

`Auto`

- `SoloA`, que representa un auto solo sin nada enganchado atrás, se asume por defecto que tiene 2 ejes y que su altura es de 1.8m.
- `ConTrailer`, que es un auto con algo enganchado atrás. Si el auto es con trailer se debe pasar como parámetro la cantidad de ejes y la altura del trailer. `Ejes` y `Altura` serán los sinónimos de tipos (`Int` y `Float` respectivamente) que representen dichas magnitudes.
- Este tipo representa las dos versiones en que puede aparecer un auto por la casilla de peajes.

`Rodado`

- Este tipo representa todos los móviles que pueden llegar a la cabina, se identifican las posibilidades enumeradas a continuación.
- `Moto`, representa una moto, no se guardan más datos acerca de la moto y se considera por defecto que no tiene ejes y que su altura es de 1.0m.
- `Automovil`, representa la llegada de un auto a la casilla pero como tiene dos versiones deberá estar asociado a un `Auto`.
- `Bus`, representa un colectivo y se debe asociar a la cantidad de ejes y la altura que tiene, o sea deberá tener como parámetros cosas del tipo `Ejes` y `Altura`.
- `Camion`, representa un camión llegando a la casilla y se debe asociar a la cantidad de ejes, a la altura que tiene y deberá también tener el peso que puede transportar en toneladas, o sea deberá tener como parámetros cosas del tipo `Ejes, Altura y Peso` (el cual será un sinónimo del tipo `Int`).

a) Definir el tipo `Rodado` que consta de los constructores `Moto`, `Automovil`, `Bus` y `Camion`, constructores con parámetros descriptos arriba (se deben definir también los sinónimos de tipos `Ejes`, `Altura` y `Peso` como sinónimos de `Int, Float y Int` respectivamente). `También definir el tipo Auto`. **Ningún tipo de los definidos deben estar en la clase `Eq`**. Agregue la clase Show en los tipos que necesite.

b) Definir igualdad para el tipo de `Rodado`: de tal manera que dos valores de tipo Rodado son iguales sólo si son móviles con la misma cantidad de ejes o si uno tiene más ejes que

otro entonces el de mayor cantidad de ejes debe tener menos altura que el anterior. Puede crear funciones auxiliares si necesita.
NOTA: Dejar como comentario en el código dos ejemplos en los que probaste la igualdad.

c) Definir la función `autosConTrailer` de la siguiente manera:

```
autosConTrailer :: [Rodado] -> Altura -> [Rodado]
```

que dada una lista de `Rodado lr` y un valor `a` de `Altura`, devuelve la lista de rodados que aparecen en `lr` que son autos con trailer cuya altura es mayor que `a`.
NOTA: Dejar como comentario un ejemplo donde hayas probado la función `autosConTrailer` con una lista con al menos 3 elementos.

d) Definir la función que chequea si hay dos rodados consecutivos iguales en una lista de rodados, la función `dosIguales,` tiene la siguiente definición de tipos:

```
dosIguales :: [Rodado] -> Bool
```

Dada una lista de `Rodado lr,` debe devolver `True` en caso que en la lista `lr` existan dos rodados que sean iguales de manera consecutiva, y `False` en caso contrario.
NOTA: Dejar como comentario en el código dos ejemplos en los que probaste la función.

Ejercicio 2

Como parte de un programa en Haskell para una agencia de viajes, se necesita modelar por un lado un tramo de un viaje, el cual estará definido por dos ciudades y la distancia entre ellas, por otro lado un viaje está definido como un tramo único (si es un viaje directo) o como un tramo más el resto del viaje si es que es con escalas.

a) Definir un tipo recursivo `Viaje`, que permite guardar las características de cada viaje. El tipo `Viaje` tendrá dos constructores:

   1) `Unico`, con un parámetro del tipo Tramo, con este constructor se modela el viaje directo sin escalas.

   2) `ConEscala,` con dos parámetros, el primero del tipo `Tramo` y el segundo del tipo `Viaje`, con este modelamos un viaje con escalas.

   Definir también el tipo `Tramo` el cual tiene un sólo constructor llamado `DefinicionDelTramo` con 3 parámetros, los dos primeros del tipo `Ciudad` y el último del tipo `Distancia`, los cuales son sinónimos de `String` e `Int` respectivamente.

b) Programar la función `viajePasaPor` la cual es un predicado que chequea si un viaje pasa por una ciudad determinada (pasada como parámetro al momento de invocarla).

```
viajePasaPor :: Viaje -> Ciudad -> Bool
```

NOTA: Dejar como comentario un ejemplo donde hayas probado `viajePasaPor`.

c) Programar la función `tramoMasLargo` con la siguiente declaración:

```
tramoMasLargo :: Viaje -> Tramo
```

que toma una variable `v` de tipo **`Viaje`** y devuelve el tramo más largo que compone ese viaje.

NOTA: Dejar como comentario un ejemplo donde hayas probado la función.

d) Agregar el tipo `Tramo` a la clase `Eq` donde la condición de igualdad es que dos tramos son iguales si contiene las mismas ciudades, notar que un tramo no tiene direccionalidad.

NOTA: Dejar como comentario un ejemplo donde hayas probado la función.

RECORDAR: Todo código entregado debe compilar y debe ajustarse a la especificación del problema.