Parcial 1 - Algoritmos I Taller: Tema F

Debés entregar el código completo en los campos correspondientes de cada ejercicio del formulario en el que completaste tus datos personales. Este código debe poder ejecutarse en haskell sin errores. Te recomendamos para ello que pruebes con diferentes ejemplos antes de entregar.

Descripción del Problema

En este examen se plantea la necesidad de representar trenes de carga, en el que los vagones pueden ser de diferentes tipos, como vagones de minerales (pueden transportar carbón, cobre, hierro, etc) y vagones cisterna (que transportan líquidos como petroleo, agua, oxigeno liquido, etc). Cada tipo de vagón tiene características específicas que se tiene en cuenta en sus constructores.

Ejercicio 1

a)

Definir el tipo de dato `Vagon` que consta de dos constructores, `Minerales` y `Cisterna` que tienen los siguiente parámetros:

- El constructor `Minerales`, toma como parámetros el nombre del mineral que transporta, la capacidad de carga en toneladas del vagón y la cantidad de toneladas que transporta.
- El constructor `Cisterna`, que toma como parámetros el líquido que transporta, la cantidad de litros transportados (medido en miles de litros), la temperatura de operación en grados centígrados (la temperatura a la que se mantiene el vagón).

b)

A partir del tipo definido en el punto anterior, definir las siguientes constantes:

- ```
  vagonCobre :: Vagon
  vagonCobre = <COMPLETAR>
  ```

  que se corresponde a un vagón de minerales que transporta `"Cobre"`, que tiene capacidad de `50` toneladas y que lleva una carga de `35` toneladas.

- ```
  vagonAgua :: Vagon
  vagonAgua = <COMPLETAR>
  ```

que se corresponde a un vagón cisterna que transporta `"Agua"`, en cantidad de `25` mil litros y que tiene una temperatura de operación de `15.8` grados.

c)

Definir la función `excesoMinerales :: Vagon -> Int -> Bool` que dado un vagón y un valor máximo de peso en toneladas n, verifica si el vagón de minerales transporta más n toneladas de minerales.

d)

Definir la función `cisternaFrio :: Vagon -> Bool` que verifica si un vagón cisterna tiene un valor de temperatura de funcionamiento inferior a 5 grados centígrados, en cualquier otro caso debe devolverse `False`.

e)

Definir la función `minimaCarga :: [Vagon] -> Int` que dada una lista `xs` de vagones, debe devolver la cantidad de toneladas transportadas por el vagón de minerales que menos peso transporta de `xs`. Tener en cuenta para el caso base que se considera que la carga más grande de un vagón de minerales es de `100` toneladas.

Ejercicio 2

Dado el tipo recursivo `Tren` definido de la siguiente manera

```
data Tren = SinVagones | Encadena Vagon Tren
```

podemos definir `ejemploTren` de la siguiente manera

```
ejemploTren :: Tren
ejemploTren = Encadena vagonAgua (Encadena vagonCobre SinVagones)
```

Definir la función `sumarCarga :: Tren -> Int -> Tren` que dado un tren `ts` y un número de toneladas `k` devuelve un nuevo tren donde a los vagones de `ts` que transportan minerales se les suma la cantidad `k` a la carga que llevan, siempre y cuando la suma sea menor o igual a la capacidad del vagón (caso contrario los deja igual que en `ts`). A los vagones cisterna los deja sin modificar.