Parcial 1 - Algoritmos I Taller: Tema H

Debes 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 las billeteras virtuales normalmente encontramos descuentos y reintegros. Vamos a diseñar un sistema que nos permita tener funciones para ver qué billetera virtual usar para ahorrar más dinero.

Ejercicio 1:

a)

Definir el tipo Ahorro que consta de dos constructores `Descuento` y `Reintegro` con los siguientes parámetros:

- El constructor `Descuento` debe tomar como parámetros el nombre de la billetera (por ejemplo "PersonalPay", "Uala", "BnaMas", "NaranjaX"), la categoría, es decir en qué es el descuento (por ejemplo "Combustible", "Supermercado", "ComidaRapida"), el monto de descuento, el mes de descuento (expresado numéricamente).
- El constructor `Reintegro` debe tomar como parámetros el nombre de la billetera virtual en que te darán reintegro, el concepto y el monto de reintegro.

Para las billeteras y los descuentos en particular, se pueden crear tipos de datos nuevos o simplemente usar Strings e Int cuando corresponda.

b)

A partir del tipo definido en el punto anterior, definí los siguientes ejemplos de Ahorro:

```
supermercadoUa :: Ahorro
supermercadoUa = <COMPLETAR>
```

correspondiente al descuento “Uala” en categoría de “Supermercado”, de $1500, para el mes 10 (octubre).

```
premioBna :: Ahorro
premioBna = <COMPLETAR>
```

correspondiente al reintegro “BnaMas”, en “premio cumplidor”, por $3000.

c)

Definir la función esReintegro `:: Ahorro -> String -> Bool` que dado ahorro y el nombre de una billetera b, devuelve `True` si el ahorro es un reintegro (notar que no puede ser desceunto) correspondiente a la billetera b, False en caso contrario.

d)

Definir la función `esDescuentoPrimavera :: Ahorro -> Bool` que dado un `Ahorro`, devuelve `True` si el ahorro es un descuento de los meses de septiembre, octubre y noviembre.

e)

Definir la función `totalDesceuntosMes :: [Ahorro] -> Int -> Int` que dada una lista de de ahorros y un mes, devuelve la suma de los descuentos del mes dado.

Ejercicio 2:

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

```
data ColaAhorro = NoHayAhorro | AgregarAhorro Ahorro ColaAhorro deriving Show
```

podemos definir los ahorros de Juan como

```
ahorrosJuan :: ColaAhorro

ahorrosJuan = AgregarAhorro supermercadoUa (AgregarAhorro premioBna
NoHayAhorro)
```

Definir la función `buenosAhorros :: ColaAhorro -> ColaAhorro`
que dada una cola de ahorros q, devuelve la cola de ahorros (descuentos o reintegros) de montos superiores a los $5000 (en el mismo orden que aparecen en q).