Parcial 1 - Algoritmos I Taller: Tema C

Ejercicio 1

En las siguientes preguntas marque la respuesta correcta.

a) Si tengo una función con la siguiente declaración
`f :: (a -> b) -> [a] -> [b]` puedo decir que:
   1) f podría ser la función map.
   2) f podría ser la función filter.
   3) Es un función polimórfica Ad hoc
   4) Es una función recursiva
   5) ninguna de las anteriores.

b) Se puede decir del tipo Maybe a:
   1) Es un tipo recursivo.
   2) Es un tipo paramétrico.
   3) Agrega un elemento a los ya representados por a.
   4) 1) y 3) son correctas.
   5) 2) y 3) son correctas.

c) Si tengo la siguiente definición de un tipo y una función, puedo decir que:
```
data Radio = AM | FM | SW
esFM :: Radio -> Bool
esFM (AM) = (AM /= FM)
```
   1) No puedo usar igualdad, debería usar pattern matching.
   2) No puedo usar pattern matching, debería usar igualdad.
   3) La función no cubre todos los casos
   4) 1 y 3 son correctas
   5) Ninguna de las anteriores

d) El mecanismo de pattern matching :
   1) Se puede usar sólo en funciones recursivas.
   2) Se puede usar sólo en tipos recursivos.
   3) Se puede usar sólo en tipos algebraicos.
   4) Se puede usar siempre.
   5) Ninguna de las anteriores.

Ejercicio 2

Se va a representar el stock de una casa de electrodomésticos usando tipos en Haskell. Los dispositivos que tenemos en cuenta son: Televisores, Home Theater . La idea es poder detallar para cada tipo de electrodoméstico, las características más importantes. En tal sentido identificamos las siguientes características:

`Televisor`

- `PulgadasTele`, que es un tipo enumerado con las siguientes opciones: `TreintayDos`, `Cuarenta`, `Setenta`
- `TipoPantalla`, que es un tipo enumerado con las siguientes opciones: `QLED`,`OLED`
- `Precio`, que es un sinónimo de `Int` indicando el precio

`HomeTheater`

- `Potencia`, que es un sinónimo de `Int` indicando la potencia de salida
- `TipoDeSalida`, que es un tipo enumerado con las siguientes opciones: `Mono`, `Estereo`, `CincoPUno`
- `Precio`, que es un sinónimo de `Int` indicando el precio

Para ello:

a) Definir el tipo `Dispositivo` que consta de los constructores `Televisor` y `HomeTheater`, con los constructores con parámetros descritos arriba (Se deben definir también los tipos enumerados `PulgadasTele`,`TipoPantalla y TipoDeSalida`). **Los tipos `Dispositivo y PulgadasTele` no deben estar en la clase `Eq`**, ni en la clase **`Ord`**.

b) Definir igualdad para el tipo de `Dispositivo` : de tal manera que, dos dispositivos de tipo `Televisor` son iguales sólo si tienen el mismo **tipo de pantalla** y las mismas **pulgadas**, mientras que dos `HomeTheater` son iguales sólo si tienen la misma **potencia**. Como es de suponer los Televisores y HomeTheater son distintos.
NOTA: Dejar como comentario en el código dos ejemplos en los que probaste la igualdad.

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

```
cuantosTelevisores :: [Dispositivo] -> TipoPantalla -> Int
```

que dada una lista de `Dispositivos ld` y un valor t de `TipoPantalla`, me devuelve un entero indicando la cantidad de televisores que hay en `ld` con tipo de pantalla t.
NOTA: Dejar como comentario un ejemplo donde hayas probado la función `cuantosTelevisores` con una lista con al menos 3 `Dispositivo`.

d) Definir la función, no hay dos dispositivos iguales de manera consecutiva en una lista de dispositivos. La función noHay`DosIguales,` tiene la siguiente definición de tipos:

```
noHayDosIguales :: [Dispositivo] -> Bool
```

Dada una lista de `Dispositivos ld`, debe devolver `True` en caso que en `ld` no existan dos dispositivos que sean iguales de manera consecutiva en la lista, y `False` en caso contrario.
NOTA: Dejar como comentario en el código dos ejemplos en los que probaste la función.

Ejercicio 3

Queremos hacer un programa, para que el profe de una materia pueda computar la situación de regularidades y promociones al final del cursado.

a) Definir un tipo recursivo `NotasDelCuatri`, que permite guardar las notas que tuvo cada alumno en el año. El tipo `NotasDelCuatri`, tendrá dos constructores:

1) `NotasDelAlumno`, que tiene 4 parámetros:

- `String`, para el nombre y apellido del alumno
- `Int` (con la nota del primer Parcial, entre 1 y 10)
- `Int` (con la nota del segundo Parcial, entre 1 y 10)
- `Int` (con la nota del recuperatorio 1 a 10,)
- `NotasDelCuatri,` recursión con el resto de las notas.

2) `NoHayMasNotas`, que es un constructor sin parámetros, similar al de la lista vacía, para indicar que se terminaron las notas.

- La condición de regularidad al finalizar el cuatrimestre puede ser, `Regular`, `Promocional` o `Libre`. A continuación se define la condición final en base a las notas obtenidas:
- Para ser considerado **`Regular`**, es necesario sacar 6 o más en cada parcial, o haber aprobado alguno de los dos parciales y haber aprobado el recuperatorio.
- Para ser considerado **`Promocional`**, es necesario haber aprobado ambos parciales y tener un promedio mayor o igual a 8 entre esas notas.
- Los alumnos son considerados con condición **`Libre`**, cuando reprobaron (menos de 6) ambos parciales, o aprobaron solo uno y luego reprobaron el recuperatorio.

Para reflejar esta situación, tienes que definir el tipo `Condicion` con los constructores `Regular`, `Libre`, `Promocional` .

b) Programar la función `esLibreAlumno`, que toma como primer parámetro `notas` del tipo **`NotasDelCuatri,`** y como segundo parámetro toma `nombre` de tipo **`String`**,

retorna un valor de tipo **`Bool`**, indicando que si el alumno con `nombre` es **libre** o no (si el alumno no está será considerado libre).

```
esLibreAlumno :: NotasDelCuatri -> String -> Bool
```

NOTA: Dejar como comentario en el un ejemplo donde hayas probado `esRegularAlumno` con un parámetro de tipo **`NotasDelCuatri`** que tenga al menos 3 alumnos.

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

```
devolverPromedio :: NotasDelCuatri -> String -> Maybe Float
```

que toma una variable notas de tipo `NotasDelCuatri`, y como segundo argumento un `nombre`, que identifica el alumno, y en caso que el alumno esté en `notas`, retorna el promedio entero (usar div) que tiene ese alumno y Nothing en caso contrario, el promedio deberá tener en cuenta que si el alumno aprobó los dos parciales entonces no se usa la nota del recuperatorio y si tuvo que hacer recuperatorio entonces el parcial desaprobado es reemplazado por el recuperatorio para el promedio.

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