Recuperatorio Haskell - Algoritmos I Taller

Ejercicio 1

Programar la función:

```
suma_multiplos :: [Int] -> Int -> Int
```

que dada una lista `xs` de enteros y un valor entero `m` devuelve la suma de los elementos de `xs` que son múltiplos de `m`.

La función se debe programar por composición, **sin recursión**, eligiendo una de las

siguiente alternativas:

- Usando `filter` y la función `sumatoria` definida en el Proyecto 1, cuyo tipado es:
  ```
  sumatoria :: [Int] -> Int
  ```
- Usando `sumatoria'` definida en el Proyecto 1, cuyo tipado es:
  ```
  sumatoria' :: [a] -> (a -> Int) -> Int
  ```

En ambos casos se pueden definir las funciones auxiliares que se consideren necesarias.

En caso de no poder resolverlo por composición, se puede dar una definición recursiva, **pero el ejercicio sumará menos puntos en ese caso.**

Ejercicio 2

Se va a construir una base de datos de estudiantes de la facultad. Para ello se deben definir los tipos de datos:

- Tipo `Carrera`: Es un tipo que tiene cuatro constructores sin parámetros que son `Matematica`, `Astronomia`, `Fisica` y `Computacion`.
- Tipo `Nombre`: Es un sinónimo del tipo `String`
- Tipo `Legajo`: Es un sinónimo de `Int`

IMPORTANTE: El tipo `Carrera` no debe pertenecer a la clase `Eq`. Si no se cumple este requisito el ejercicio no suma puntos.

Finalmente se debe definir el tipo `Estudiante` con un único constructor `Est` que toma tres parámetros: el primero de tipo `Legajo`, el segundo de tipo `Nombre` y el tercero de tipo `Carrera`.

a) Programar utilizando recursión y *pattern matching* la función:

```
buscar :: [Estudiante] -> Carrera -> [Nombre]
```

que dada una lista de estudiantes `xs` y una carrera `c`, devuelve los nombres de los estudiantes en `xs` que cursan la carrera `c`.

b) Inventar un ejemplo concreto con una lista de al menos 3 elementos, ejecutarlo y escribirlo como comentario en lo que se suba al parcial.

Ejercicio 3

Utilizando el tipo *lista de asociaciones* definido como:

```
data ListaAsoc a b = Vacia | Nodo a b (ListaAsoc a b)
```

a) Programar usando recursión la función:

```
la_existe :: ListaAsoc a b -> a -> Bool
```

que dada una lista de asociaciones `la` y una clave `k` indica si dicha clave está en la lista de asociaciones `la`. Completar el tipado de la función incluyendo al tipo `a` en las clases que se necesiten para poder programarla.

b) Inventar un ejemplo concreto con una lista de al menos 3 elementos, ejecutarlo y escribirlo como comentario en lo que se suba al parcial.