Parcial 1 - Algoritmos I Taller: Tema C

Ejercicio 1

Se van a implementar algunos aspectos de una tienda de ropa en Haskell.

a) Definir el tipo `Ropa` que consta de los constructores `Camisa`, `Pantalon`, `Pollera`, `Short`. Los constructores no toman parámetros. **El tipo `Ropa` no debe estar en la clase `Eq`**. Luego programa la función usando *pattern matching*:

```
misma_ropa :: Ropa -> Ropa -> Bool
```

que dados dos valores `p1` y `p2` del tipo `Ropa` debe devolver `True` cuando `p1` y `p2` son la misma ropa (se construyen con el mismo constructor) y `False` en caso contrario.

Si se usan más de cinco casos, o menos de cinco casos este apartado sumará menos puntaje.

b) Definir el tipo `Prenda` que representa una prenda de ropa. Tiene constructores:

- Constructor `ConTalle`: Toma dos parámetros, el primero de tipo `Talle` y el segundo de tipo `Ropa`
- Constructor `TalleUnico`: Toma un sólo parámetro de tipo `Ropa`

El tipo `Talle` debe ser un sinónimo del tipo `Int`.

c) Programar la función

```
valor_talle :: Prenda -> Int
```

teniendo en cuenta que el valor de una prenda será:

- Si es una prenda con talle : Su valor es el talle de la prenda.
- Si es una prenda talle único : Su valor es 0.

d) Incluir el tipo `Prenda` en la clase `Ord` de manera tal que una prenda se considere mayor que otra si su valor según la función `valor_talle` es más grande.

Ejercicio 2

a) Programar de manera recursiva la función

```
solo_con_talle :: [Prenda] -> Ropa -> [Talle]
```

que dada una lista de prendas `ps` y una ropa `r` devuelve una lista con los números de los talles de las prendas con talle (las que no son talle único) de `ps` que son de la ropa `r`.

b) Escribir una lista de prendas con al menos tres elementos, donde al menos uno de ellos debe tener talle, y otro debe ser talle único.

c) Escribir el resultado de `solo_con_talle` para la lista del punto b)

Ejercicio 3

Basados en el tipo `ListaAsoc` del *Proyecto 2*, programar la función:

```
la_duplica_pares :: ListaAsoc a b -> ListaAsoc a b
```

que dada una lista de asociaciones `la,` devuelve una nueva lista de asociaciones con las asociaciones de `la` cuyos valores son `: Si la clave es par el valor será multiplicado por 2, y si la clave es impar permanecerá el valor que tenía.` Completar el tipado de la función para incluir los *type clases* necesarios para programarla.

Ejercicio 4*

a) Programar la función

```
a_esCota_inf :: a -> Arbol a -> Bool
```

que dado un valor `e` de tipo `a` y un árbol `as` indica si `e` es una cota inferior de todos los elementos dentro del árbol `as`. Es decir indica si `e` es menor o igual a todos los elementos del árbol `as`. Completar el tipado de la función para incluir los *type classes* necesarios para programarla

b) Inventar un ejemplo de uso de la función creando un árbol con al menos 3 elementos

c) Escribir el resultado de la función aplicada al ejemplo del inciso b)