Fundamentos de Computación
Entregable 5
Listas

Este trabajo tiene un puntaje de 9 puntos y debe ser realizado en forma **INDIVIDUAL**.
Se debe subir a Aulas un archivo .hs con las funciones antes del día **10/11/21 a las 21 hs**.
Se pueden utilizar **todas las funciones del preludio vistas en clase** para listas, naturales y booleanos**,** y **todos los lemas de los repartidos del curso.**

Ejercicio 1

1. Defina la función **remDups :: Eq a => [a] → [a]**, que recibe una lista y elimina todas las repeticiones de elementos de la misma.
Ejemplo: **remDups [1,3,2,5,2,6,1,1,2] = [1,3,2,5,6]** (en este o cualquier otro orden)**.**

```
remDups = \l → case l of {
                        [ ] -> .....
                      ; x:xs -> x : (filter ..... (remDups xs) )}
```

2. Demuestre que **(∀l::[a]) remDups (remDups l) = remDups l**
   Puede utilizar el siguiente resultado sin necesidad de demostrarlo:
   Lema$_{filter}$: (∀p::a→Bool) (∀l::[a]) remDups (filter p l) = filter p (remDups l).

3. Demuestre que **(∀l::[a]) length (remDups l) ≤ length l**

Ejercicio 2

Introducimos el tipo de los *conjuntos* de elementos de tipo a, de la siguiente manera:

type Set a = [a]

Se pide implementar las siguientes funciones sobre conjuntos,

NOTA: en las funciones que reciben listas que representan conjuntos se puede asumir que éstas no contienen elementos repetidos, y el resultado también debe ser una lista sin elementos repetidos

1. Defina, sin usar funciones auxiliares, la función **pertenece :: Eq a => a → Set a → Bool,** que verifica si un elemento pertenece a un conjunto.

Ejemplos:
pertenece 42 [1,3,2,5,6] = False
pertenece False [True,False] = True

2. Defina con recursión la función **incluido:: Eq a => Set a → Set a → Bool,** que verifica si el primer conjunto está incluido en el segundo (sin importar el orden de los elementos).
Ejemplos:
incluido [3,1,2] [1,2,3,4] = True
incluido [True, False] [False] = False
incluido [[], [1,2,3], [4,5]] [[], [1,3], [4,5]] = False

3. Defina la función **iguales** :: **Eq a => Set a → Set a → Bool**, que determina si dos conjuntos tienen los mismos elementos (sin importar el orden de sus elementos).
Ejemplos:
Iguales [3,1,2] [1,2,3] = True
Iguales [4,6] [4,5,6] = False

4. Defina con recursión la función **union:: Eq a => Set a → Set a → Set a**, que calcula la unión de dos conjuntos.

5. Defina con recursión la función **interseccion:: Eq a => Set a → Set a → Set a**, que calcula la intersección de dos conjuntos.

Ejercicio 3 (Opcional = 2 puntos extra)

Redefina las funciones 2, 4 y 5 del ejercicio 2 en una sola línea, sin utilizar recursión, y usando como funciones auxiliares las funciones de listas vistas en clase.
Nombre a las funciones redefinidas igual que las originales, pero con un “2” al final (por ejemplo: **interseccion2 :: Eq a => Set a → Set a → Set a**).