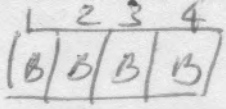

UBA – Facultad de Ciencias Exactas y Naturales – Departamento de Computación– Taller de Álgebra I
Parcial– Mayo 2016

LU o DNI:
Apellidos:
Nombres:

Aclaraciones: Para aprobar se requiere al menos tener 2 ejercicios bien resueltos.

Ejercicio 1
Programe la función `sumatoriaPares :: [Integer] -> Integer` que devuelve la suma de los elementos pares de la lista. Por ejemplo:
`sumatoriaPares [1,2,3,4]` $\rightsquigarrow$ 6.

Ejercicio 2
Programe la función filtro, que quita todas las apariciones de un elemento en una lista. Por ejemplo:
`filtro [1,2,3] 3` $\rightsquigarrow$ `[1,2]`
`filtro [1,2] 3` $\rightsquigarrow$ `[1,2]`

Ejercicio 3
Programe la función `posiciones :: Integer -> [Integer] -> Integer`, tal que `posiciones x ys` devuelve todas las posiciones del elemento x que están en la lista ys, y devuelve`[]` en el caso de que `x` no pertenezca a la lista. Las posiciones pueden ser devueltas en cualquier orden. Por ejemplo:
`posiciones 5 [1,5,3,5,6,5,3,4]` $\rightsquigarrow$ `[2,4,6] o [6,4,2]`
`posiciones 7 [1,5,3,5,6,5,3,4]` $\rightsquigarrow$ `[]`

Ejercicio 4
Programe la función `inserta :: Integer -> [Integer] -> [Integer]`, que dado un número `e` y una lista `l` ordenada de menor a mayor, agrega `e` de forma ordenada, es decir, agrega el número `e` delante del primer elemento que sea mayor o igual que él.
Luego, programe la función `ordenaPorInserción :: [Integer] -> [Integer]`, que ordena una lista mediante una insercion. Por ejemplo:
`ordenaPorInserción [2,4,3,6,3]` $\rightsquigarrow$ `[2,3,3,4,6]`

Ejercicio 1

```
sumatoriaPares :: [Integer] -> Integer
sumatoriaPares lista
  | length lista == 0 = 0
  | (head lista) `mod` 2 == 0 = head lista +
    sumatoriaPares (tail lista)
  | otherwise = sumatoriaPares (tail lista)
```

Ejercicio 2

```
filtro :: [Integer] -> Integer -> [Integer]
filtro lista elem
  | length lista == 0 = []
  | head lista == elem = filtro (tail lista) elem
  | head lista /= elem = (head lista) : filtro (tail lista) elem
```

ACLARACIÓN: Programé esta función sólo para enteros porque tuve problemas con el operador == y las funciones con asignación de tipo genérico (creo que GHCI se quejaba de que no podía usarlo). Siempre "solucioné" estos bugs restringiendo la función a Integer o Float según fuera adecuado. El algoritmo, más allá de estos problemas, debería funcionar con filtro :: [a] -> a -> [a], pero no sé cómo son las particularidades de la comparación de elementos no numéricos en Haskell.

Ejercicio 3

```
juntarIndices :: Integer -> Integer -> [Integer] -> [Integer]
juntarIndices indice elem lista
  | length lista == 0 = []
  | (head (lista) == elem = indice :
    juntarIndices (indice +1) elem (tail (lista)
  | (head lista) /= elem = juntarIndices (indice +1) elem (tail (lista)

posiciones: Integer -> [Integer] -> [Integer]
posiciones elem lista = juntarIndices 1 elem lista
```

Ejercicio 4

```
inserta :: Integer -> [Integer] -> [Integer]
inserta elem lista
  | length lista == 0 = elem:lista
  | elem > (head lista) = (head lista): (inserta elem (tail lista))
  | otherwise = elem : lista

ordenaPorInsercion :: [Integer] -> [Integer]
ordenaPorInsercion lista
  | length lista == 0 = []
  | otherwise = inserta (head lista) (ordenaPorInsercion (tail lista))
```