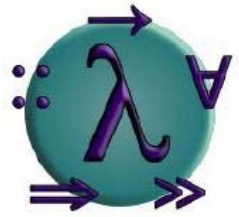


## Practico N° 8 – Listas

### Notas:

- Previo a la realización de este práctico es necesario leer el siguiente material sobre: - [Listas o Secuencias definidas sobre algún conjunto dado](#)



Recuerda, que antes de definir las siguientes funciones debes importar el [Módulo con la definición inductiva del conjunto N \(Naturales\)](#) utilizando el siguiente texto en las primeras líneas del archivo:

Las primeras dos líneas de Practico7.hs serán:

```
module Practico7 where
```

```
import Naturales
```

--siguen las declaraciones de tipos, funciones, etc

**Observación:** Aquí sí es importante que el nombre del módulo, que coincide con el del archivo .hs que lo contiene se escriba en mayúscula y sin la extensión .hs

1. Definir en Haskell las siguientes funciones:

- a) **aNat** que dado un Integer devuelva su representación en el conjunto N.  
Ejemplo:  $\text{aNat } 4 = (\text{S } (\text{S } (\text{S } (\text{S } 0))))$
- b) **deNat** que dado un elemento de N devuelva su representación en el conjunto Integer.  $\text{deNat } (\text{S } (\text{S } 0)) = 2$

2. Definir la función **igList** que toma dos listas de enteros e indica si son iguales.

3. Definir las siguientes funciones por análisis de casos.

- a) **esVacía**, que dada una secuencia devuelve True si se trata de la secuencia vacía y False en otro caso.
- b) **concatenar**, que dadas dos secuencias devuelve la secuencia resultado de agregar todos los elementos de la segunda a continuación de los de la primera.

4. El predicado **ordenada** que dada una lista de enteros especifica si la lista está ordenada en forma creciente según la relación  $< : \mathbb{Z} \rightarrow \mathbb{Z} \rightarrow \text{Bool}$ .

5. Definir las siguientes funciones por casos:

- a) **sumaList**, que dada una secuencia de naturales devuelve la suma de todos los naturales que pertenecen a ella si la secuencia es vacía devuelve 0.
- b) **multiList**, como sumaList, pero devuelve el producto de todos los naturales.
- c) **invertir**, recibe una secuencia y devuelve la secuencia resultado de invertir la secuencia dada.
- d) **tomar\_n**, recibe un natural n y una secuencia y devuelve la secuencia formada por el resultado de tomar de la secuencia dada los n primeros elementos. Para la secuencia nil, siempre devuelve nil.  
Ejemplos:  $(\text{tomar\_n } 3 [2,3,4,5,1]) = [2,3,4]$   
 $(\text{tomar\_n } 3 [1,1]) = [1,1]$
- e) **borrar\_n**, similar a tomar, pero devuelve el resultado de eliminar los n primer elementos de la secuencia dada.  
Ejemplos :  $(\text{borrar\_n } 3 [2,3,4,5,1]) = [5,1]$   
 $(\text{borrar\_n } 3 [1,1]) = \text{nil}$

- f) **posicion**, dada una secuencia y un elemento devuelve la posición del elemento en la secuencia. Si el elemento no pertenece a la secuencia devuelve 0.
- g) **mapear**, recibe una función de  $Z$  en  $Z$  y una secuencia de enteros. Devuelve una secuencia de enteros que es el resultado de aplicar la función dada a cada uno de los elementos de la secuencia original.  
Ejemplo: (mapear factorial [2,3,4,0]) = [2,6,24,1]
- h) **filtrar**, recibe una función de  $Z$  en Bool y una secuencia de enteros y devuelve la secuencia que tiene únicamente los elementos de la secuencia original que para los cuales la aplicación de la función recibida es Verdadera.  
Ejemplo: (filtrar es\_par [2,3,3,5,4,6]) = [2,4,6]
- i) **existe**, recibe una secuencia de elementos de un cierto conjunto  $A$  y una función  $f:A \rightarrow \text{Bool}$  y devuelve un valor del conjunto Bool. Devuelve True si hay por lo menos un elemento  $x$  en la lista que verifica  $f$ , en otro caso devuelve False.
- j) **pertenece**, que recibe un elemento y una lista y devuelve True si el elemento dado está en la lista, False en caso contrario.
- k) **borrar**, recibe una lista y un elemento  $n$ , y devuelve la lista resultado de borrar todas las ocurrencias de  $n$  en la lista original. Si el elemento no ocurre en la secuencia dada, la devuelve tal cual, si ocurre más de una vez, borra todas las ocurrencias.
- l) **insertaOrd**, recibe una lista de enteros ordenada en forma creciente y un entero  $e$  inserta el entero en la lista de forma que el resultado siga siendo una lista ordenada.
- m) **multiplicar**, recibe un número y una secuencia de naturales. Devuelve la secuencia resultado de multiplicar todos los números de la secuencia original por el número dado. Se pide que se defina utilizando mapear y la función multiplicación en  $N$ , en su versión prefija elaborada en el práctico mult:  $N \rightarrow N \rightarrow N$ .
- n) **capicua**, recibe una secuencia y devuelve un valor del conjunto bool. Devuelve True si y solo si la secuencia recibida es capicúa.  
Ejemplos: (capicua [2,3,5,6,5,3,2]) = True  
(capicua [a, b, b, a, b]) = False  
(capicua []) = True

6. Definir todas las funciones pedidas anteriormente en Haskell.

### **Ejercicios complementarios sobre Listas**

7. Dada una lista de enteros (Integer), devolver el elemento mayor de la lista.
8. Dada una lista de pares ordenados de enteros, devolver una lista con el primer elemento de cada par.
9. Dada una lista de pares ordenados de enteros, devolver una lista con el elemento mayor de cada uno de los pares.
10. Dada una lista de enteros devolver la lista con los elementos pares de la lista dada.
11. Dada una lista de naturales, devolver un natural que sea la suma de los naturales de la lista que son impares.

12. Dada una lista y un natural  $n$ , devolver el elemento que ocupa la posición  $n$  en la lista, si  $n$  es mayor que el largo de la lista, devuelve un mensaje de error. Se considera que el primer elemento de la lista ocupa la posición 0.
13. Dada un alista de enteros devolver el menor entero de la lista que sea múltiplo de 3.
14. Dada una lista de enteros devolver la lista que se conforma a partir de la dada, sin elementos repetidos.
15. Dado un String, indicar si contiene alguna vocal.
16. Dado un String y un caracter, devolver el String resultado de eliminar todas las ocurrencias del caracter dado en él.
17. Dado un numero natural  $n$ , devolver una lista con todos los divisores de  $n$ .
18. Dada una lista de enteros, devolver la lista ordenada de menor a mayor.
19. Dada una lista de listas de algún tipo, devolver una lista que sea el resultado de la concatenación de todas las listas de la lista dada.
20. Dada una función binaria y dos listas, devuelve una lista resultado de aplicar la función a los elementos de las listas dadas, tomados de dos en dos  
Ejemplo:  
Hugs> fun20 (+) [1...3] [10, 11, 12, 13, 14, 15, 16]  
[11, 13, 15]
21. Definir una función que se comporte similar a la anterior, pero que construya una lista de pares a partir de las dos listas.  
Ejemplo:  
Hugs> fun21 [1...3] ['a', 'b', 'c', 'd', 'e']  
[(1, 'a'), (2, 'b'), (3, 'c')]