FUNDAMENTOS DE COMPUTACIÓN
PRÁCTICO 4
LISTAS

En este práctio nos dedicatemos a demostrar propiedades de las funciones definidas en el laboratorio de listas.

Ej.1. Pruebe las siguientes propiedades de la concatenación de listas:

1) `(∀l::[a]) l++[] = l`
2) `(∀l,l2::[a]) length (l1++l2) = length l1 + length l2`
3) Asociatividad del `++`.

Ej.2. Pruebe las siguientes propiedades de la función `reverse`:

1) `(∀x::a) reverse [x] = [x]`
2) `(∀l1,l2::[a]) reverse(l1++l2) = (reverse l2)++(reverse l1)`
3) `(∀l::[a]) reverse (reverse l) = l`

Ej.3. Pruebe las siguientes propiedades de las funciones definidas en el laboratorio de listas:

1) `(∀l::[a],∀f::(a->b)) length (map f l) = length l`
2) `(∀l::[a]) sum (map doble l) = doble (sum l)`.
   Podrá usar sin demostrar las propiedades de la suma y el siguiente resultado:
   `(∀x,y::N) doble (x + y) = doble x + doble y`
3) `(∀l::[[a]]) length (concat l) = lensum l`
4) `(∀l::[a]) l = pares (duplicar l)`
5) `(∀p::(a->Bool))(∀l::[a]) length (filter p l)` $\leq$ `length l`
6) `(∀p::(a->Bool))(∀l::[a]) length (filter p l) = cuantos p l`
7) `(∀p::(a->Bool))(∀l::[a]) all p (filter p l) = True`.
8) `(∀xs::[a]) length (sinRep xs)` $\leq$ `length xs`

Puede hacer uso de los lemas de $\leq$ vistos en clase.

Ej.4.

1) Defina la función `int` que recibe dos listas y retorna una lista que contiene los miembros de ambas intercalados. Si las listas fueran de distinto largo el intercalado se completará con los elementos de la lista más larga.
2) Demuestre que:
   `(∀l1::[a])(∀l2::[a]) length (int l1 l2) = length l1 + length l2`.
   Se podrá utilizar sin demostrar la asociatividad y conmutatividad de $+$.

Ej.5.

1) Defina la función `contar` que dados una lista y un elemento retorne la cantidad de ocurrencias del elemento en la lista. ¿Qué tipo tiene esta función? Justifique.
2) Defina, sin usar funciones auxiliares, la función `generar::N -> N -> [Bool]` que dados dos enteros `m` y `n`, genere una lista de booleanos conteniendo `m` ocurrencias de `True` seguidas de `n` ocurrencias de `False`.
   Ejemplo: `generar 3 2 = [True, True, True, False, False]`.
3) Demuestre que `(∀m,n::N) contar (generar m n) True = m`.

Ej.6.

1) Defina la función `numeros::N -> [N]` que recibe un natural `n` y genera una lista conteniendo los números `[n,n-1,..,0]`.
2) Defina la función `maxL::[N] -> N` que recibe una lista de naturales y devuelve el máximo de la misma. Para la lista vacía deberá devolverse como resultado a `0`.
3) Demuestre que `(∀n::N) maxL (numeros n) = n`.
   Puede hacer uso de los lemas de $\leq$ vistos en clase.

Ej.7.

1) Defina una función `foldr` que generalice el esquema de recursión primitiva para listas. ¿Cuál es el tipo de `foldr`?
2) Utilizando `foldr` defina las funciones `sum1, prod1, and1, or1` y `concat1`, equivalentes a las definidas en el laboratorio.
3) Pruebe que su definción de `sum` usando `foldr` es extensionalmente igual su definición original: `(∀l::[a]) sum l = sum1 l`