Práctica 7: Sobre Funciones

Funciones de Alto Orden

Definimos nuestras propias versiones de `map`, `reduce` y `filter`:

```
-fun simpleMap (F,nil) = nil
   | simpleMap(F,x::xs) = F(x)::simpleMap(F,xs);
val simpleMap = fn : ('a ->'b) * 'a list ->'b list
-exception ListaVacia;
exception ListaVacia
-fun reduce (F,nil) = raise ListaVacia
   | reduce(F,[a]) = a
   | reduce(F,x::xs) = F(x, reduce(F,xs));
val reduce = fn : ('a * 'a ->'a) * 'a list ->'a
-fun filter (P,nil) = nil
   | filter(P,x::xs) = if P(x) then x::filter(P,xs) else filter(P,xs);
val filter = fn : ('a ->bool) * 'a list ->'a list
```

Teniendo en cuenta estas definiciones, defina las funciones que se le solicitan en los siguientes ejercicios:

1. Defina dos funciones, una sin usar y otra usando la definición de `simpleMap`, para cada uno de los items siguientes:

1. Defina funciones que dada una lista de reales devuelvan otra lista con los cuadrados de la lista pasada como argumento.
2. Defina funciones que dada una lista de enteros devuelvan otra lista con los cubos de la lista pasada como argumento.
3. Defina funciones que dada una lista de reales devuelva otra lista donde cada elemento negativo sea reemplazado por el cero, dejando los no negativos sin cambiar.
4. Defina funciones que dada una lista de enteros devuelva otra lista donde todos los elementos de la nueva lista correspondan a los elementos de la lista pasada como argumento aumentados en una unidad.
5. Defina funciones que dada una lista de caracteres devuelva otra lista donde cada caracter en minúscula es reemplazado por su correspondiente en mayúscula, el resto no varía.

2. Defina dos funciones, una sin usar y otra usando la definición de `reduce`, para cada uno de los items siguientes:

1. Defina funciones que tome una lista de reales y devuelva el máximo.
2. Defina funciones que tome una lista de reales y devuevla el mínimo.
3. Defina funciones que calculen el `OR` de una lista de booleanos.

3. Defina dos funciones, una sin usar y otra usando la definición de `filter`, para cada uno de los items siguientes:

1. Defina funciones que dada una lista de reales devuelva otra con los elementos mayores a 0.
2. Defina funciones que dada una lista de reales devuelva otra con los elementos que se encuentren entre 1 y 2.
3. Defina funciones que dada una lista de strings devuelva otra que contenga únicamente los strings que comienzan con el caracter `#"a"`.
4. Defina funciones qeu dada una lista de strings devuelva otra que contenga únicamente los strings cuya longitud es como máximo 3.

Funciones Currificadas

4. Escriba una versión currificada para cada una de las funciones que se le pide a continuación. Defina:

1. Una función que toma una lista de enteros y un número entero `n` y devuelve `true` si el resultado la suma de todos los elementos de la lista es mayor a `n` o `false` en caso contrario
2. Una función que toma una lista de enteros y un número entero `n` y devuelve otra lista con los elementos de la primera lista elevados a la potencia `n`
3. Una función que tome una lista de funciones y un valor y aplique cada función al valor, devolviendo una lista con los resultados.

Combinando `simpleMap`, `reduce` y `filter`

5. Escriba las siguientes funciones:

1. Una función `suma_cubos` que dada una lista de números `list`, devuelva la suma de los cubos de los números positivos de `list`.
2. Una función `long_lists` que toma una lista de listas y devuelve `true` si y sólo si las longitudes de todas las sublistas son mayores a 4.
3. Una función `prod_positivo` que dada una lista `list` compuesta por listas de números devuelva otra lista que contenga únicamente las sublistas de `list` las cuales al multiplicar todos sus elementos obtengamos un resultado positivo.