Paquete #3 de ejercicios (recursividad)

1) Resuelva de nuevo los **15** ejercicios del paquete anterior pero ahora en forma recursiva.

16) Defina una función recursiva ***Find*** que reciba dos argumentos: ***elem*** y ***lista***. La función debe devolver ***NIL*** si ***elem*** *no* es un elemento de ***lista***, de lo contrario, deberá devolver la sublista que comienza con la primera instancia de ***elem***.

17) Defina una función recursiva ***Cambia*** que reciba como argumento una ***lista*** y dos elementos ***elem1***, ***elem2***. Como respuesta, la función debe entregar otra lista parecida a la original, pero donde todas las ocurrencias de ***elem1*** se substituyeron por ***elem2***.

18) En el URL http://www.cliki.net/fibonacci se presentan diversas implementaciones para los números de Fibonacci. Implemente **TODAS** las opciones que ahí se presentan y compare su desempeño con ***time*** para el argumento **50.**

19) Defina una función recursiva ***Mapea*** que opere exactamente igual que la función ***mapcar*** de *Common Lisp*.

20) Defina una función recursiva ***Aplana*** que reciba como argumento una lista con elementos anidados a cualquier nivel de profundidad y, como respuesta, entregue una lista conteniendo los mismos elementos pero todos ellos al nivel principal de profundidad.

21) Defina una función recursiva ***Elimina*** que reciba como argumento una ***lista*** y un número real ***n***. La función debe entregar como resultado una copia de la lista original, en la cual se hayan eliminado todos los elementos que no sean numéricos, así como todos aquellos elementos numéricos que sean menores o iguales que ***n***.

22) Defina una función recursiva ***PegaYCambia*** que reciba como argumento dos listas ***lista1*** ***lista2*** y dos elementos ***elem1***, ***elem2***. Como respuesta, la función debe entregar una lista donde concatene las dos listas originales, pero substituyendo todas las ocurrencias (en ambas listas) de ***elem1*** por ***elem2***.

23) Defina una función ***QSort*** que reciba como argumento único una ***lista*** e implemente con ellos el algoritmo de ordenamiento ***Quick Sort***, ignorando por completo aquellos elementos de la lista original que no sean numéricos. La respuesta de la función debe ser una lista con los elementos numéricos de la original ordenados de forma ascendente.