Paradigmas de Programación

Práctica 5

1. Redefina en un fichero ***mylist3.ml*** las siguientes funciones de modo que no se utilice recursividad no terminal.

```
let rec suml = function
    [] -> 0
  | h::t -> h + suml t;;

let rec maxl = function
    [] -> raise (Failure "maxl")
  | h::[] -> h
  | h::t -> max h (maxl t);;

let rec to0from n =
    if n < 0 then []
    else n :: to0from (n-1);;

let rec fromto m n =
    if m > n then []
    else m :: fromto (m+1) n;;

let rec from1to n =
    if n < 1 then []
    else from1to (n-1) @ [n];;

let append = List.append;;

let concat = List.concat;;

let map = List.map;;

let power x y =
    let rec innerpower x y =
        if y = 0 then 1
        else x * innerpower x (y-1)
    in
    if y >= 0 then innerpower x y
    else invalid_arg "power";;

let fib n =
    let rec innerfib n =
        if n < 2 then n
        else innerfib (n-1) + innerfib (n-2)
    in
    if n >= 0 then innerfib n
    else invalid_arg "fib";;

let fact n =
    let rec innerfact n =
        if n = 0 then 1.
        else float n *. innerfact (n-1)
    in
    if n >= 0 then innerfact n
    else invalid_arg "fact";;

let incseg l = List.fold_right (fun x t -> x::List.map ((+) x) t) l [];;

let rec multicomp l x = match l with
    [] -> x
  | f::t -> f (multicomp t x);;

let rec insert x = function
        [] -> [x]
      | h::t -> if x <= h then x::h::t
                else h :: insert x t;;
```

```
let rec insert_gen f x l = match l with
        [] -> [x]
      | h::t -> if f x h then x::l
                else h :: insert_gen f x t;;
```

2. [Ejercicio opcional] Implemente en un fichero ***hanoi.ml*** una función

```
hanoi : 'a * 'a * 'a -> int -> ('a * 'a) list
```

que resuelva el problema de *Las Torres de Hanoi*. Los tres valores de tipo `'a` permiten nombrar las torres con valores de cualquier tipo. El valor de tipo `int` indica el número de discos a mover desde la primera torre a la tercera. La lista de pares `('a * 'a)` indica los movimientos individuales de cada disco en formato (torre origen, torre destino). Ejemplos de ejecución:

```
# hanoi (1,2,3) 3;;
- : (int * int) list =
[(1, 3); (1, 2); (3, 2); (1, 3); (2, 1); (2, 3); (1, 3)]

# hanoi ("A","B","C") 4;;
- : (string * string) list =
[("A", "B"); ("A", "C"); ("B", "C"); ("A", "B"); ("C", "A"); ("C", "B");
 ("A", "B"); ("A", "C"); ("B", "C"); ("B", "A"); ("C", "A"); ("B", "C");
 ("A", "B"); ("A", "C"); ("B", "C")]
```

3. [Ejercicio opcional] Implemente en un fichero ***lprod.ml*** una versión terminal de la función

```
lprod : 'a list -> 'b list -> ('a * 'b) list
```

descrita en el enunciado de práctica 4.