2.- Ejercicios básicos a resolver

Ejercicio 2.1. Escribir una función recursiva que devuelva el sumatorio desde un valor entero hasta otro.

```
***> Ejercicio 2.1
red eval(
  let rec 'suma(x,y) =
    if x >= y then y else x + 'suma(x + 1, y)
  in 'suma(3,7)
) .
```

Ejercicio 2.2. Define una función binaria que devuelva el mayor de sus dos argumentos.

```
***> Ejercicio 2.2
red eval(
  let 'mayor (x, y) = if x >= y
                      then x
                      else y
  in 'mayor (100, 1)
) .
```

Ejercicio 2.3. Define una función que calcule el factorial de un número.

```
***> Ejercicio 2.3
red eval(
  let rec 'facto (x) = if x <= 1
                       then 1
                       else x * 'facto(x - 1)
  in 'facto (3)
) .
```

3.- Ejercicios a resolver sobre tipos lista

Ejercicio 3.1.Define una función para contar cuántas veces aparece un elemento en una lista

```
***> Ejercicio 3.1
red eval(
  let rec 'contList(x,y) = if null?(x)
                           then 0
                           else if car(x) == y
                             then 1 + 'contList(cdr(x),y)
                             else 'contList(cdr(x),y)
  in 'contList([1,2,2,4,5,6,2,2],2)
) .
```

Ejercicio 3.2. Define una función que reemplace en una lista todas las ocurrencias de un elemento n por otro elemento p.

```
***> Ejercicio 3.2
red eval(
  let rec 'remList(x,y,z) = if null?(x)
                            then []
                            else if car(x) == y
                              then cons(z,'remList(cdr(x),y,z))
                              else cons(car(x),'remList(cdr(x),y,z))
  in 'remList([1,2,3,4,4,3,4,3],4,7)
) .
```

5. Ejercicios a resolver sobre paso por referencia

Ejercicio 5.1.Define una función binaria que intercambie los valores de sus dos argumentos

```
***> Ejercicio 5.1
red eval(
  let x = 3 and y = 25 and z = 0 and 'swap(x,y,z) = {
    * z := * y ; * y := * x ; * x := * z ; cons(* x, cons(* y, []))
  }
  in 'swap(& x, & y , & z)
) .

```

Ejercicio 5.2. Define una función que extraiga el valor mínimo y máximo de una lista de enteros usando paso por referencia.

```
***> Ejercicio 5.2
red eval(
  let m = 9999 and x = 0 in let rec 'inter(m,x,z) = {
  if null?(z)
  then cons(* m, cons(* x , []))
  else { if(car(z)) >= * x
         then * x := car(z)
         else if car(z) <= * m
           then * m := car(z)
           else car(z) ;
         'inter(m,x,cdr(z))
  }}
  in 'inter(& m, & x, [3,4,7,4,6,0,9,5])
) .
```