Paradigmas de Programación

Práctica 6

Ejercicio 1. La "falsa función" ***rlist*** (definida a continuación) sirve para generar listas aleatorias de números enteros.

```
let rec rlist r n =
  if n <= 0 then []
  else Random.int r :: rlist r (n-1);;
```

Implemente una versión recursiva terminal de ***rlist*** con nombre ***rlist_t***.

Ejercicio 2. La función ***insert*** (definida a continuación) sirve para insertar un elemento en una lista ordenada, en el lugar que le corresponde.

La función ***isort*** (definida también a continuación) sirve para ordenar una lista mediante el método de inserción.

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

let rec isort = function
  [] -> []
| h::t -> insert h (isort t);;
```

Implemente versiones terminales de ***insert*** e ***isort*** con nombres ***insert_t*** e ***isort_t***, respectivamente.

Ejercicio 3. La función ***divide*** (definida a continuación) sirve para repartir equitativamente los elementos de una lista en dos listas.

La función ***merge*** (definida también a continuación) sirve para reunir en una lista los elementos de dos listas ordenadas, respetando su relación de orden.

La función ***msort*** (definida también a continuación) sirve para ordenar una lista mediante el método de fusión.

```
let rec divide = function
  h1::h2::t -> let t1,t2 = divide t in
               h1::t1, h2::t2
| l -> l,[];;

let rec merge l1 l2 = match l1,l2 with
  [], l | l, [] -> l
| h1::t1, h2::t2 -> if h1 <= h2 then h1::merge t1 l2
                    else h2::merge l1 t2;;
```

```
let rec msort l = match l with
  [] | [_] -> l
| _ -> let l1,l2 = divide l in
       merge (msort l1) (msort l2);;
```

Implemente versiones terminales de ***divide*** y ***merge*** con nombres ***divide_t*** y ***merge_t***, respectivamente.

Implemente con nombre ***msort_qt*** una función de ordenación de listas por fusión, que utilice ***merge_t*** y ***divide_t***, en vez de ***merge*** y ***divide***.

Ejercicio 4. La función ***qsort*** (definida a continuación) sirve para ordenar una lista mediante el método *quick-sort*.

```
let rec qsort l = match l with
  [] | [_] -> l
| h::t -> let l1,l2 = List.partition ((<=) h) t in
          qsort l2 @ (h::qsort l1);;
```

Implemente con nombre ***qsort_qt*** una función de ordenación de listas mediante el método *quick-sort*, utilizando sólo funciones definidas de modo recursivo terminal (aunque la propia definición recursiva de ***qsort_qt*** no sea necesariamente recursiva terminal).

Ejercicio 5. Utilizando la función ***rlist_t***, la función ***fromto*** implementada en la práctica 5, y la función ***crono*** (definida a continuación), compare el rendimiento de las funciones ***isort***, ***isort_t***, ***msort***, ***msort_qt***, ***qsort*** y ***qsort_qt***, indicando ventajas e inconvenientes de cada una de ellas.

```
let crono f x =
  let t = Sys.time () in
  let _ = f x in
  Sys.time () -. t;;
```

Sugerencia: Realice las mediciones utilizando listas de números enteros, ordenadas y no ordenadas, de tamaño 2, 4, 8, 16, 32, 64, 128, 256 y 512 mil elementos. Compruebe si la complejidad computacional teórica de cada algoritmo de ordenación coincide con los resultados empíricos obtenidos. Reflexione sobre el hecho de que no sea necesario que la implementación de algunas de las funciones propuestas sea recursiva terminal, para que se puedan aplicar igualmente a listas tan grandes.

Ejercicio 6. Siguiendo las definiciones de las funciones ***isort***, ***isort_t***, ***msort***, ***msort_qt***, ***qsort*** y ***qsort_qt***, desarrolle versiones correspondientes para cada una de ellas, con nombres ***isort_gen***, ***isort_t_gen***, ***msort_gen***, ***msort_qt_gen***, ***qsort_gen*** y ***qsort_qt_gen***, que incorporen el orden como argumento, de modo que tengan tipo ***('a -> 'a -> bool) -> 'a list -> 'a list***.

Nota importante. Realice en un fichero ***sort.ml*** las implementaciones citadas en los ejercicios 1, 2, 3, 4 y 6. Escriba en un fichero ***sort.txt*** el estudio comparativo y las conclusiones alcanzadas respecto a las cuestiones planteadas en el ejercicio 5.