```
 1 open List;;
 2
 3 (* implementazione del tipo di dato 't seq *)
 4 type 't seq = 't list;; (* sequenze polimorfe implementate con liste *)
 5
 6 (* operazioni di 't seq richieste in QuickSort *)
 7 let (size:'t seq ->int) =
 8             (* Out: numero di elementi 't contenuti in seq *)
 9             (* effect: nessuna modifica                    *)
10         fun x -> length x
11 ;;
12 let (pivot:'t seq -> 't) =
13             (* Out: ElementoSelezionato come pivot *)
14             (* effect: nessuna modifica            *)
15         fun x -> hd x
16 ;;
17 let rec (triPartition: 't seq -> ('t -> 't -> bool) -> 't
18                         -> ('t seq * 't seq * 't seq)) =
19             (* Out: partiziona seq (rispetto a pv, con ordine op) in una *)
20             (*         tripla (Minori,Uguali,Maggiori)                   *)
21             (* effect: nessuna modifica                                  *)
22         fun seq op pv ->
23             match seq with
24                 |[] -> ([],[],[])
25                 |x::xs -> let ((lt:'t seq),(eq:'t seq),(gt:'t seq))
26                                 = triPartition xs op pv in
27                           if op x pv then (x::lt,eq,gt)
28                           else if op pv x then (lt,eq,x::gt)
29                           else (lt,x::eq,gt)
30 ;;
31 let (concatena: 't seq -> 't seq -> 't seq) =
32             (* Out: concatena seq1 con seq2 *)
33             (* effect: nessuna modifica     *)
34         fun x y -> x@y
35 ;;
36 let (cons: 't -> 't seq -> 't seq) =
37             (* Out: aggiunge x in testa ad xs *)
38             (* effect: nessuna modifica       *)
39         fun x xs -> x::xs
40 ;;
41
42
```