```
 1 (* implementazione del tipo di dato 't seq *)
 2 type 't seq = Nil | Cons of 't * ('t seq);;    (* Tipo di Dato Algebrico *)
 3
 4 (* exceptions *)
 5 exception Err1 of string * string;;
 6
 7 (* operazioni di 't seq richieste in QuickSort *)
 8 let rec size seq = match seq with
 9             (* Out: numero di elementi 't contenuti in seq *)
10             (* effect: nessuna modifica                    *)
11         | Nil -> 0
12         | Cons(_,xs) -> 1 + size xs
13 ;;
14 let pivot seq = match seq with
15             (* Out: coppia (ElementoSelezionato,SequenzaRimanente) per seq *)
16             (* effect: nessuna modifica                                    *)
17         | Nil -> raise (Err1("pivot", "illegal arg: Nil seq"))
18         | Cons(x,xs) -> (x,xs)
19 ;;
20 let rec dividi seq op pv = match seq with
21             (* Out: partiziona seq (rispetto a pv, con ordine op) in una *)
22             (*         coppia (MinoriUguali,Maggiori) *)
23             (* effect: nessuna modifica               *)
24         | Nil -> (Nil,Nil)
25         | Cons(x,xs) ->
26                let (le,gt) = dividi xs op pv in
27                if (op x pv)||(x = pv) then ((Cons(x,le)),gt)
28                else (le,Cons(x,gt))
29         ;;
30 let rec concatena seq1 seq2 = match seq1 with
31             (* Out: concatena seq1 con seq2 *)
32             (* effect: nessuna modifica     *)
33         | Nil -> seq2;
34         | Cons(x,xs) -> Cons(x,(concatena xs seq2))
35 ;;
36 let cons x xs =
37             (* Out: aggiunge x in testa ad xs *)
38             (* effect: nessuna modifica       *)
39         Cons(x,xs)
40 ;;
41
42
```