```
1: ::::::::::::::
2: qsort.ml
3: ::::::::::::::
4: (* $Id: qsort.ml,v 361.2 2006-03-03 14:36:37-08 - - $ *)
5:
6: (*
7: * Tricky version of qsort.
8: * Note:  O(n^2) if pivot is badly chosen.
9: *)
10:
11: let filter = List.filter ;;
12: let compose f g x = f (g x) ;;
13:
14: let rec qsort (<?) list = match list with
15:     | [] -> []
16:     | car::[] -> list
17:     | car::cdr ->
18:       let (large) = filter ((<?) car) cdr
19:       and (small) = filter (compose not ((<?) car)) cdr
20:       in qsort (<?) small @ [car] @ qsort (<?) large
21:     ;;
22:
23: let thelist = [6; 7; 11; 8; 4; 2; 9; -4; 10] ;;
24:
25: qsort (<) thelist ;;
26: qsort (>) thelist ;;
27:
28: ::::::::::::::
29: qsort.ml.script
30: ::::::::::::::
31: bash-1$ ocaml
32:         OCaml version 4.02.1
33:
34: # #use "qsort.ml";;
35: val filter : ('a -> bool) -> 'a list -> 'a list = <fun>
36: val compose : ('a -> 'b) -> ('c -> 'a) -> 'c -> 'b = <fun>
37: val qsort : ('a -> 'a -> bool) -> 'a list -> 'a list = <fun>
38: val thelist : int list = [6; 7; 11; 8; 4; 2; 9; -4; 10]
39: - : int list = [-4; 2; 4; 6; 7; 8; 9; 10; 11]
40: - : int list = [11; 10; 9; 8; 7; 6; 4; 2; -4]
41: #
```