```
 1: (* $Id: msorttailrec.ml,v 341.4 2014-11-17 14:05:54-08 - - $ *)
 2:
 3: (*
 4: * A better version of mergesort.
 5: * Uses tail recursion for split and merge, but msort is O(log n) deep.
 6: * Note that in this case split reverses the list.
 7: * Merge then reverses it again giving properly sorted final output,
 8: * but msort has to alternate between less and not less on alternate
 9: * levels of the recursion.
10: * The functions split and merge should probably be nested inside
11: * msort, but they are left external for easier debugging.
12: * Also given here explicitly are higher order functions.
13: *)
14:
15: let rec foldl fn ident list = match list with
16:     | []       -> ident
17:     | car::cdr -> foldl fn (fn ident car) cdr
18:
19: let rec foldr fn ident list = match list with
20:     | []       -> ident
21:     | car::cdr -> fn car (foldr fn ident cdr)
22:
23: let cons car cdr = car::cdr
24:
25: let swap fn x y = fn y x
26:
27: let revcat = foldl (swap cons)
28:
29: let reverse = revcat []
30:
31: let un boolfn x y = not (boolfn x y)
32:
```

```
33:
34: let merge less list1 list2 =
35:     let rec merge' in1 in2 out = match (in1, in2) with
36:             | ([], [])   -> out
37:             | ([], list) -> revcat out list
38:             | (list, []) -> revcat out list
39:             | (car1::cdr1 as list1), (car2::cdr2 as list2)
40:                          -> if less car2 car1
41:                             then merge' cdr1 list2 (car1::out)
42:                             else merge' list1 cdr2 (car2::out)
43:     in  merge' list1 list2 []
44:
45: let split list =
46:     let rec split' list out1 out2 = match list with
47:         | []              -> (out1, out2)
48:         | [car]           -> (car::out1, out2)
49:         | car::cadr::cddr -> split' cddr (car::out1) (cadr::out2)
50:     in split' list [] []
51:
52: let msort less list =
53:     let rec msort' less list = match list with
54:         | []            -> []
55:         | [car] as list -> list
56:         | list          -> let (list1, list2) = split list
57:                            in  merge less (msort' (un less) list1)
58:                                           (msort' (un less) list2)
59:     in msort' less list
60:
61: let msortlt = msort (<)
62:
63: ;;
64:
65: msortlt [33;11;-10;12;44;202;8;66];;
66:
```

```
    1: bash-1$ ocaml
    2:         OCaml version 4.02.1
    3:
    4: # #use "msorttailrec.ml";;
    5: val foldl : ('a -> 'b -> 'a) -> 'a -> 'b list -> 'a = <fun>
    6: val foldr : ('a -> 'b -> 'b) -> 'b -> 'a list -> 'b = <fun>
    7: val cons : 'a -> 'a list -> 'a list = <fun>
    8: val swap : ('a -> 'b -> 'c) -> 'b -> 'a -> 'c = <fun>
    9: val revcat : '_a list -> '_a list -> '_a list = <fun>
   10: val reverse : '_a list -> '_a list = <fun>
   11: val un : ('a -> 'b -> bool) -> 'a -> 'b -> bool = <fun>
   12: val merge : ('_a -> '_a -> bool) -> '_a list -> '_a list -> '_a list = <
fun>
   13: val split : 'a list -> 'a list * 'a list = <fun>
   14: val msort : ('_a -> '_a -> bool) -> '_a list -> '_a list = <fun>
   15: val msortlt : '_a list -> '_a list = <fun>
   16: - : int list = [-10; 8; 11; 12; 33; 44; 66; 202]
   17: # exit 0;;
   18:
```