Tik-79.144 **Syksy 1997**
Logiikka tietotekniikassa: perusteet
Laskuharjoitus 10
Ratkaisut

1. `partition`-predikaatin tehtävänä on osittaa lukulista kahtia annetun luvun perusteella. Toiseen listaan tulee annettua lukua pienemmät tai yhtäsuuret luvut ja toiseen listaan annettua lukua suuremmat luvut (lista käydään lineaarisesti lävitse). Predikaatti `qsort` järjestää luvut kutsumalla `partition`-predikaattia ja järjestämällä syntyneet osalistat rekursiivisilla `qsort`-kutsuilla. Lopputulos muodostetaan `append`-predikatin avulla kytkemällä järjestetyt osalistat ja osituksessa käytetty luku (syötteenä saadun listan ensimmäinen luku) oikeaan järjestykseen. Alla on esitetty debuggerilla saadut tulosteet (a)- ja (b)-kohtien kyselyistä, joiden avulla voi seurata tulkin suorittaman resoluution etenemistä. Huomaa, että laskennan välituloksissa esiintyy muuttujia (nimi muotoa `_<numero>`), jotka on otettu käyttöön uudelleenimettäessä ohjelman klausuulien muuttujia.

a) Kutsu osittaa lukulistan [1,3] 2:ta pienempiin tai yhtäsuuriin ja suurempiin lukuihin, siis listoiksi [1] ja [3].

```
| ?- partition(2,cons(1,cons(3,nil)),X,Y).
 + 1  1  Call: partition(2,cons(1,cons(3,nil)),_141,_157) ?
   2  2  Call: 1=<2 ?
   2  2  Exit: 1=<2 ?
 + 3  2  Call: partition(2,cons(3,nil),_341,_157) ?
   4  3  Call: 3=<2 ?
   4  3  Fail: 3=<2 ?
   4  3  Call: 3>2 ?
   4  3  Exit: 3>2 ?
 + 5  3  Call: partition(2,nil,_341,_655) ?
 + 5  3  Exit: partition(2,nil,nil,nil) ?
 + 3  2  Exit: partition(2,cons(3,nil),nil,cons(3,nil)) ?
 + 1  1  Exit: partition(2,cons(1,cons(3,nil)),
                         cons(1,nil),cons(3,nil)) ?

X = cons(1,nil),
Y = cons(3,nil) ?

yes
```

b) Kutsu järjestää lukulistan [4,2,1,3] lukulistaksi [1,2,3,4].

```
| ?- qsort(cons(4,cons(2,cons(1,cons(3,nil)))),X).
 + 1  1  Call: qsort(cons(4,cons(2,cons(1,cons(3,nil)))),
                     _179) ?
 + 2  2  Call: partition(4,cons(2,cons(1,cons(3,nil))),
                         _389,_390) ?
   3  3  Call: 2=<4 ?
   3  3  Exit: 2=<4 ?
 + 4  3  Call: partition(4,cons(1,cons(3,nil)),_532,_390) ?
   5  4  Call: 1=<4 ?
   5  4  Exit: 1=<4 ?
 + 6  4  Call: partition(4,cons(3,nil),_851,_390) ?
   7  5  Call: 3=<4 ?
   7  5  Exit: 3=<4 ?
 + 8  5  Call: partition(4,nil,_1165,_390) ?
 + 8  5  Exit: partition(4,nil,nil,nil) ?
 + 6  4  Exit: partition(4,cons(3,nil),cons(3,nil),nil) ?
 + 4  3  Exit: partition(4,cons(1,cons(3,nil)),
                         cons(1,cons(3,nil)),nil) ?
 + 2  2  Exit: partition(4,cons(2,cons(1,cons(3,nil))),
                           cons(2,cons(1,cons(3,nil))),nil) ?
 + 9  2  Call: qsort(cons(2,cons(1,cons(3,nil))),_396) ?
 + 10  3  Call: partition(2,cons(1,cons(3,nil)),_1968,_1969) ?
   11  4  Call: 1=<2 ?
   11  4  Exit: 1=<2 ?
 + 12  4  Call: partition(2,cons(3,nil),_2106,_1969) ?
   13  5  Call: 3=<2 ?
   13  5  Fail: 3=<2 ?
   13  5  Call: 3>2 ?
   13  5  Exit: 3>2 ?
 + 14  5  Call: partition(2,nil,_2106,_2420) ?
 + 14  5  Exit: partition(2,nil,nil,nil) ?
 + 12  4  Exit: partition(2,cons(3,nil),nil,cons(3,nil)) ?
 + 10  3  Exit: partition(2,cons(1,cons(3,nil)),
                          cons(1,nil),cons(3,nil)) ?
 + 15  3  Call: qsort(cons(1,nil),_1975) ?
 + 15  3  Exit: qsort(cons(1,nil),cons(1,nil)) ?
 + 16  3  Call: qsort(cons(3,nil),_1981) ?
 + 16  3  Exit: qsort(cons(3,nil),cons(3,nil)) ?
 + 17  3  Call: append(cons(1,nil),cons(2,cons(3,nil)),_396) ?
```

```
 + 18  4  Call: append(nil,cons(2,cons(3,nil)),_3506) ?
 + 18  4  Exit: append(nil,cons(2,cons(3,nil)),
                       cons(2,cons(3,nil))) ?
 + 17  3  Exit: append(cons(1,nil),cons(2,cons(3,nil)),
                       cons(1,cons(2,cons(3,nil)))) ?
 + 9  2  Exit: qsort(cons(2,cons(1,cons(3,nil))),
                     cons(1,cons(2,cons(3,nil)))) ?
 + 19  2  Call: qsort(nil,_402) ?
 + 19  2  Exit: qsort(nil,nil) ?
 + 20  2  Call: append(cons(1,cons(2,cons(3,nil))),cons(4,nil),
                       _179) ?
 + 21  3  Call: append(cons(2,cons(3,nil)),cons(4,nil),_4240) ?
 + 22  4  Call: append(cons(3,nil),cons(4,nil),_4366) ?
 + 23  5  Call: append(nil,cons(4,nil),_4487) ?
 + 23  5  Exit: append(nil,cons(4,nil),cons(4,nil)) ?
 + 22  4  Exit: append(cons(3,nil),cons(4,nil),
                       cons(3,cons(4,nil))) ?
 + 21  3  Exit: append(cons(2,cons(3,nil)),cons(4,nil),
                       cons(2,cons(3,cons(4,nil)))) ?
 + 20  2  Exit: append(cons(1,cons(2,cons(3,nil))),cons(4,nil),
                       cons(1,cons(2,cons(3,cons(4,...))))) ?
 + 1  1  Exit: qsort(cons(4,cons(2,cons(1,cons(3,nil)))),
                     cons(1,cons(2,cons(3,cons(4,nil))))) ?

X = cons(1,cons(2,cons(3,cons(4,nil)))) ?

yes
```

2. a)
```
qmove(X1,Y1,X2,Y2) :- diff(X1,X2,_), equal(Y1,Y2).
qmove(X1,Y1,X2,Y2) :- diff(Y1,Y2,_), equal(X1,X2).
qmove(X1,Y1,X2,Y2) :- diff(X1,X2,N), diff(Y1,Y2,N).

equal(1,1).
equal(X,Y) :- succ(A,X), succ(B,Y), equal(A,B).

diff(X,Y,N) :- add(X,N,Y).
diff(X,Y,N) :- add(Y,N,X).

add(X,1,Y) :- succ(X,Y).
add(X,Y,Z) :- succ(A,Y), succ(B,Z), add(X,A,B).

succ(1,2).
succ(2,3).
succ(3,4).
succ(4,5).
succ(5,6).
succ(6,7).
succ(7,8).
```

b)
```
nqmove(X1,Y1,X2,Y2) :- diff(X1,X2,N), diff(Y1,Y2,M),
                       diff(N,M,_).
```

c)
```
nocc(X,Y,[]) :- equal(X,Y).
nocc(X,Y,[]) :- diff(X,Y,_).
nocc(X1,Y1,[pos(X2,Y2)|L]) :- nqmove(X1,Y1,X2,Y2),
                              nocc(X1,Y1,L).

noccupied(X,8,[]) :- nocc(X,8,[]).
noccupied(X1,Y1,[pos(X2,Y2)|L]) :- succ(Y1,Y2),
  noccupied(X2,Y2,L), nocc(X1,Y1,[pos(X2,Y2)|L]).

solution([pos(X,1)|L]) :- noccupied(X,1,L).
```