```
#|

 Copyright (C) 1994 by Yuan Yu.  All Rights Reserved.

 This script is hereby placed in the public domain, and therefore unlimited
 editing and redistribution is permitted.

 NO WARRANTY

 Yuan Yu PROVIDES ABSOLUTELY NO WARRANTY.  THE EVENT SCRIPT IS PROVIDED "AS IS"
 WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT
 LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
 PARTICULAR PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE
 SCRIPT IS WITH YOU.  SHOULD THE SCRIPT PROVE DEFECTIVE, YOU ASSUME THE COST OF
 ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

 IN NO EVENT WILL Yuan Yu BE LIABLE TO YOU FOR ANY DAMAGES, ANY LOST PROFITS,
 LOST MONIES, OR OTHER SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT
 OF THE USE OR INABILITY TO USE THIS SCRIPT (INCLUDING BUT NOT LIMITED TO LOSS
 OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY THIRD
 PARTIES), EVEN IF YOU HAVE ADVISED US OF THE POSSIBILITY OF SUCH DAMAGES, OR
 FOR ANY CLAIM BY ANY OTHER PARTY.

|#
```

EVENT: Start with the library "mc20-2" using the compiled version.

```
;           Proof of the Correctness of a Quicksort Program
;
#|
The following C function QSORT sorts a[left], ..., a[right] into increasing
order.  The program is a slightly modified version of K&R.

/* qsort: sort a[left]...a[right] into increasing order.  We use the middle */
/* element of each subarray for partitioning.   */
void qsort (int a[], int left, int right)
{
  int i, last, temp;

  if (left >= right)
    return;
  last = (left + right) / 2;
  temp = a[left];
```

```
  a[left] = a[last];
  a[last] = temp;
  last = left;
  for (i = left + 1; i<= right; i++)
    if (a[i] < a[left]){
      temp = a[++last];
      a[last] = a[i];
      a[i] = temp;
    };
  temp = a[left];
  a[left] = a[last];
  a[last] = temp;
  qsort(a, left, last-1);
  qsort(a, last+1, right);
}
```

Here is the MC68020 assembly code of the above QSORT program. The code is generated by "gcc -O".

```
0x22b8 <qsort>:         linkw fp,#0
0x22bc <qsort+4>:       moveml d2-d4/a2-a3,sp@-
0x22c0 <qsort+8>:       moveal fp@(8),a3
0x22c4 <qsort+12>:      movel fp@(12),d3
0x22c8 <qsort+16>:      movel fp@(16),d4
0x22cc <qsort+20>:      cmpl d3,d4
0x22ce <qsort+22>:      ble 0x2338 <qsort+128>
0x22d0 <qsort+24>:      movel d3,d2
0x22d2 <qsort+26>:      addl d4,d2
0x22d4 <qsort+28>:      bpl 0x22d8 <qsort+32>
0x22d6 <qsort+30>:      addql #1,d2
0x22d8 <qsort+32>:      asrl #1,d2
0x22da <qsort+34>:      movel 0(a3)[d3.l*4],d1
0x22de <qsort+38>:      movel 0(a3)[d2.l*4],0(a3)[d3.l*4]
0x22e4 <qsort+44>:      movel d1,0(a3)[d2.l*4]
0x22e8 <qsort+48>:      movel d3,d2
0x22ea <qsort+50>:      movel d2,d0
0x22ec <qsort+52>:      bra 0x2308 <qsort+80>
0x22ee <qsort+54>:      moveal 0(a3)[d0.l*4],a0
0x22f2 <qsort+58>:      cmpal 0(a3)[d3.l*4],a0
0x22f6 <qsort+62>:      bge 0x2308 <qsort+80>
0x22f8 <qsort+64>:      addql #1,d2
0x22fa <qsort+66>:      movel 0(a3)[d2.l*4],d1
0x22fe <qsort+70>:      movel 0(a3)[d0.l*4],0(a3)[d2.l*4]
0x2304 <qsort+76>:      movel d1,0(a3)[d0.l*4]
```

```
0x2308 <qsort+80>:      addql #1,d0
0x230a <qsort+82>:      cmpl d0,d4
0x230c <qsort+84>:      bge 0x22ee <qsort+54>
0x230e <qsort+86>:      movel 0(a3)[d3.l*4],d1
0x2312 <qsort+90>:      movel 0(a3)[d2.l*4],0(a3)[d3.l*4]
0x2318 <qsort+96>:      movel d1,0(a3)[d2.l*4]
0x231c <qsort+100>:     moveal d2,a0
0x231e <qsort+102>:     pea a0@(-1)
0x2322 <qsort+106>:     movel d3,sp@-
0x2324 <qsort+108>:     movel a3,sp@-
0x2326 <qsort+110>:     lea 0x22b8 <qsort>,a2
0x232a <qsort+114>:     jsr a2@
0x232c <qsort+116>:     movel d4,sp@-
0x232e <qsort+118>:     moveal d2,a0
0x2330 <qsort+120>:     pea a0@(1)
0x2334 <qsort+124>:     movel a3,sp@-
0x2336 <qsort+126>:     jsr a2@
0x2338 <qsort+128>:     moveml fp@(-20),d2-d4/a2-a3
0x233e <qsort+134>:     unlk fp
0x2340 <qsort+136>:     rts
```

The machine code of the above program is:

```
<qsort>:       0x4e56  0x0000  0x48e7  0x3830  0x266e  0x0008  0x262e  0x000c
<qsort+16>:    0x282e  0x0010  0xb883  0x6f68  0x2403  0xd484  0x6a02  0x5282
<qsort+32>:    0xe282  0x2233  0x3c00  0x27b3  0x2c00  0x3c00  0x2781  0x2c00
<qsort+48>:    0x2403  0x2002  0x601a  0x2073  0x0c00  0xb1f3  0x3c00  0x6c10
<qsort+64>:    0x5282  0x2233  0x2c00  0x27b3  0x0c00  0x2c00  0x2781  0x0c00
<qsort+80>:    0x5280  0xb880  0x6ce0  0x2233  0x3c00  0x27b3  0x2c00  0x3c00
<qsort+96>:    0x2781  0x2c00  0x2042  0x4868  0xffff  0x2f03  0x2f0b  0x45fa
<qsort+112>:   0xff90  0x4e92  0x2f04  0x2042  0x4868  0x0001  0x2f0b  0x4e92
<qsort+128>:   0x4cee  0x0c1c  0xffec  0x4e5e  0x4e75
```

In the Logic, it looks like:

```
'(78    86     0      0      72     231    56     48
  38    110    0      8      38     46     0      12
  40    46     0      16     184    131    111    104
  36    3      212    132    106    2      82     130
  226   130    34     51     60     0      39     179
  44    0      60     0      39     129    44     0
  36    3      32     2      96     26     32     115
  12    0      177    243    60     0      108    16
  82    130    34     51     44     0      39     179
```

```
  12        0         44        0         39        129       12        0
  82        128       184       128       108       224       34        51
  60        0         39        179       44        0         60        0
  39        129       44        0         32        66        72        104
  255       255       47        3         47        11        69        250
  255       144       78        146       47        4         32        66
  72        104       0         1         47        11        78        146
  76        238       12        28        255       236       78        94
  78        117)
|#

; in the Logic, the above program is defined by (qsort-code).
```

DEFINITION:
QSORT-CODE
=
```
'(78 86 0 0 72 231 56 48 38 110 0 8 38 46 0 12 40 46 0
  16 184 131 111 104 36 3 212 132 106 2 82 130 226 130
  34 51 60 0 39 179 44 0 60 0 39 129 44 0 36 3 32 2 96
  26 32 115 12 0 177 243 60 0 108 16 82 130 34 51 44 0
  39 179 12 0 44 0 39 129 12 0 82 128 184 128 108 224
  34 51 60 0 39 179 44 0 60 0 39 129 44 0 32 66 72 104
  255 255 47 3 47 11 69 250 255 144 78 146 47 4 32 66
  72 104 0 1 47 11 78 146 76 238 12 28 255 236 78 94
  78 117)
```

THEOREM: ilessp-lessp
$(x \in \mathbf{N}) \rightarrow (\text{ilessp}(x, y) = (x < y))$

EVENT: Disable ilessp.

EVENT: Enable iplus.

EVENT: Enable idifference.

EVENT: Enable iquotient.

```
; qsort is a function in Nqthm that characterizes the functional semantics of
; the program (qsort-code).
```

DEFINITION:
qpart-aux $(l, r, lst, last, i)$
= **if** $r < i$ **then** swap $(l, last, lst)$
elseif ilessp (get-nth (i, lst), get-nth (l, lst))

then qpart-aux $(l,\ r,\ \text{swap}\,(1 + \mathit{last},\ i,\ \mathit{lst}),\ 1 + \mathit{last},\ 1 + i)$
else qpart-aux $(l,\ r,\ \mathit{lst},\ \mathit{last},\ 1 + i)$ **endif**

DEFINITION:
$\text{qpart}\,(l,\ r,\ \mathit{lst}) = \text{qpart-aux}\,(l,\ r,\ \text{swap}\,(l,\ (l + r) \div 2,\ \mathit{lst}),\ l,\ 1 + l)$

DEFINITION:
qlast-aux $(l,\ r,\ \mathit{lst},\ \mathit{last},\ i)$
$=$ **if** $r < i$ **then** fix (last)
elseif ilessp (get-nth $(i,\ \mathit{lst})$, get-nth $(l,\ \mathit{lst})$)
then qlast-aux $(l,\ r,\ \text{swap}\,(1 + \mathit{last},\ i,\ \mathit{lst}),\ 1 + \mathit{last},\ 1 + i)$
else qlast-aux $(l,\ r,\ \mathit{lst},\ \mathit{last},\ 1 + i)$ **endif**

DEFINITION:
$\text{qlast}\,(l,\ r,\ \mathit{lst}) = \text{qlast-aux}\,(l,\ r,\ \text{swap}\,(l,\ (l + r) \div 2,\ \mathit{lst}),\ l,\ 1 + l)$

THEOREM: qlast-aux-lb
$(\mathit{left} \le \mathit{last}) \rightarrow (\text{qlast-aux}\,(\mathit{left},\ \mathit{right},\ \mathit{lst},\ \mathit{last},\ i) \not< \mathit{left})$

THEOREM: qlast-lb
$\text{qlast}\,(\mathit{left},\ \mathit{right},\ \mathit{lst}) \not< \mathit{left}$

THEOREM: qlast-aux-ub
$((\mathit{last} < i) \land (i \le \mathit{right}))$
$\rightarrow \quad (\mathit{right} \not< \text{qlast-aux}\,(\mathit{left},\ \mathit{right},\ \mathit{lst},\ \mathit{last},\ i))$

THEOREM: qlast-ub
$(\mathit{left} < \mathit{right}) \rightarrow (\mathit{right} \not< \text{qlast}\,(\mathit{left},\ \mathit{right},\ \mathit{lst}))$

EVENT: Disable qlast.

EVENT: Disable qpart.

DEFINITION:
qsort $(l,\ r,\ \mathit{lst})$
$=$ **if** $l < r$
then qsort $(1 + \text{qlast}\,(l,\ r,\ \mathit{lst}),$
$r,$
$\text{qsort}\,(l,\ \text{qlast}\,(l,\ r,\ \mathit{lst}) - 1,\ \text{qpart}\,(l,\ r,\ \mathit{lst})))$
else lst **endif**

```
; the computation time of the program.
```

DEFINITION:
qpart-aux-t $(a,\ l,\ r,\ n,\ lst,\ last,\ i)$
$=$ **if** $r < i$ **then** `11`
elseif ilessp (get-nth $(i,\ lst)$, get-nth $(l,\ lst)$)
then splus (`10`,
qpart-aux-t $(a,\ l,\ r,\ n,$ swap $(1 + last,\ i,\ lst),\ 1 + last,\ 1 + i))$
else splus (`6`, qpart-aux-t $(a,\ l,\ r,\ n,\ lst,\ last,\ 1 + i))$ **endif**

DEFINITION:
qpart-t $(a,\ l,\ r,\ n,\ lst)$
$=$ **let** *lst1* **be** swap $(l,\ (l + r) \div$ `2`$,\ lst)$
in
splus (`18`, qpart-aux-t $(a,\ l,\ r,\ n,\ lst1,\ l,\ 1 + l))$ **endlet**

DEFINITION: qsort-10 $(a,\ l,\ r,\ n,\ lst) =$ `10`

DEFINITION: qsort-5 $(a,\ l,\ r,\ n,\ lst) =$ `5`

DEFINITION: qsort-3 $(a,\ l,\ r,\ n,\ lst) =$ `3`

DEFINITION:
qsort-t $(a,\ l,\ r,\ n,\ lst)$
$=$ **let** *last* **be** qlast $(l,\ r,\ lst)$,
qlst **be** qpart $(l,\ r,\ lst)$
in
if $l < r$
then splus (qpart-t $(a,\ l,\ r,\ n,\ lst)$,
splus (qsort-t $(a,\ l,\ last - 1,\ n,\ qlst)$,
splus (qsort-5 $(a,\ l,\ r,\ n,\ lst)$,
splus (qsort-t $(a,$
$1 + last,$
$r,$
$n,$
qsort $(l,\ last - 1,\ qlst))$,
qsort-3 $(a,\ l,\ r,\ n,\ lst)))))$
else qsort-10 $(a,\ l,\ r,\ n,\ lst)$ **endif endlet**

`; an induction hint.`

DEFINITION:
qsort-induct $(s,\ a,\ l,\ r,\ n,\ lst)$
$=$ **let** *last* **be** qlast $(l,\ r,\ lst)$,
qlst **be** qpart $(l,\ r,\ lst)$
in
if $l < r$

then qsort-induct (stepn (s, qpart-t (a, l, r, n, lst)),
a,
l,
idifference ($last$, `1`),
n,
$qlst$)
$\wedge$ qsort-induct (stepn (s,
splus (qpart-t (a, l, r, n, lst),
splus (qsort-t (a,
l,
$last - 1$,
n,
$qlst$),
qsort-5 (a, l, r, n, lst)))),
a,
$1 + last$,
r,
n,
qsort (l, $last - 1$, $qlst$))
else t endif endlet

```
; the preconditions of the initial state.
```

DEFINITION:
qstack (l, r, lst)
$=$ **let** $last$ **be** qlast (l, r, lst),
$lst1$ **be** qpart (l, r, lst)
in
if $l < r$
then max (`40` + qstack (l, $last - 1$, $lst1$),
`52` + qstack ($1 + last$, r, qsort (l, $last - 1$, $lst1$)))
else `68` **endif endlet**

THEOREM: qstack-la0
qstack (l, r, lst) $\not<$ `68`

THEOREM: qstack-la1
let $last$ **be** qlast (l, r, lst),
$lst1$ **be** qpart (l, r, lst)
in
($l < r$)
$\rightarrow$ (qstack (l, r, lst) $\not<$ (`40` + qstack (l, $last - 1$, $lst1$))) **endlet**

THEOREM: qstack-la2
let $last$ **be** qlast (l, r, lst),

$\quad lst1$ **be** qpart $(l,\ r,\ lst)$
in
$(l < r)$
$\rightarrow \quad (\text{qstack}\,(l,\ r,\ lst)$
$\qquad \not< \quad (\texttt{52} + \text{qstack}\,(1 + last,\ r,\ \text{qsort}\,(l,\ last - 1,\ lst1)))))$ **endlet**

```
; an upper bound of stack space.
```

THEOREM: qstack-ubound-la-1
$(l < r)$
$\rightarrow \quad ((\texttt{52} * (r - l))$
$\qquad \not< \quad (\texttt{52} + (\texttt{52} * ((\text{qlast}\,(l,\ r,\ lst) - 1) - l))))$

THEOREM: qstack-ubound-la-2
$(l < r)$
$\rightarrow \quad ((\texttt{52} * (r - l))$
$\qquad \not< \quad (\texttt{52} + (\texttt{52} * (r - (1 + \text{qlast}\,(l,\ r,\ lst))))))$

THEOREM: qstack-ubound
$\text{qstack}\,(l,\ r,\ lst) \le (\texttt{68} + (\texttt{52} * (r - l)))$

EVENT: Disable qstack.

```
; the initial state.
```

DEFINITION:
qsort-statep $(s,\ a,\ l,\ r,\ n,\ lst)$
$=$ **let** sp **be** sub $(\texttt{32},\ \text{qstack}\,(l,\ r,\ lst) - \texttt{16},\ \text{read-sp}\,(s))$
in
$(\text{mc-status}\,(s) = \texttt{'running})$
$\land \quad \text{evenp}\,(\text{mc-pc}\,(s))$
$\land \quad \text{rom-addrp}\,(\text{mc-pc}\,(s),\ \text{mc-mem}\,(s),\ \texttt{138})$
$\land \quad \text{mcode-addrp}\,(\text{mc-pc}\,(s),\ \text{mc-mem}\,(s),\ \text{QSORT-CODE})$
$\land \quad \text{ram-addrp}\,(a,\ \text{mc-mem}\,(s),\ \texttt{4} * n)$
$\land \quad \text{mem-ilst}\,(\texttt{4},\ a,\ \text{mc-mem}\,(s),\ n,\ lst)$
$\land \quad \text{ram-addrp}\,(sp,\ \text{mc-mem}\,(s),\ \text{qstack}\,(l,\ r,\ lst))$
$\land \quad \text{disjoint}\,(a,\ \texttt{4} * n,\ sp,\ \text{qstack}\,(l,\ r,\ lst))$
$\land \quad (a = \text{read-mem}\,(\text{add}\,(\texttt{32},\ \text{read-sp}\,(s),\ \texttt{4}),\ \text{mc-mem}\,(s),\ \texttt{4}))$
$\land \quad (l = \text{iread-mem}\,(\text{add}\,(\texttt{32},\ \text{read-sp}\,(s),\ \texttt{8}),\ \text{mc-mem}\,(s),\ \texttt{4}))$
$\land \quad (r = \text{iread-mem}\,(\text{add}\,(\texttt{32},\ \text{read-sp}\,(s),\ \texttt{12}),\ \text{mc-mem}\,(s),\ \texttt{4}))$
$\land \quad (\text{qstack}\,(l,\ r,\ lst) < \exp\,(\texttt{2},\ \texttt{32}))$
$\land \quad (l \in \mathbf{N})$
$\land \quad (r < n)$
$\land \quad \text{uint-rangep}\,(\texttt{4} * n,\ \texttt{32})$ **endlet**

DEFINITION:
qsort-sp $(s,\ a,\ l,\ r,\ n,\ lst)$
$=$ ((mc-status (s) = `'running`)
  $\land$ evenp (mc-pc (s)))
  $\land$ rom-addrp (mc-pc (s), mc-mem (s), 138)
  $\land$ mcode-addrp (mc-pc (s), mc-mem (s), QSORT-CODE)
  $\land$ ram-addrp $(a$, mc-mem (s), $4 * n)$
  $\land$ mem-ilst (4, a, mc-mem (s), n, lst)
  $\land$ (a = read-mem (add (32, read-sp (s), 4), mc-mem (s), 4))
  $\land$ (l = iread-mem (add (32, read-sp (s), 8), mc-mem (s), 4))
  $\land$ (r = iread-mem (add (32, read-sp (s), 12), mc-mem (s), 4))
  $\land$ $(l \in \mathbf{N})$
  $\land$ $(n \in \mathbf{N})$
  $\land$ $(r < n)$
  $\land$ int-rangep $(n, 32)$
  $\land$ int-rangep $(2 * r, 32)$
  $\land$ uint-rangep $(4 * n, 32)$
  $\land$ ram-addrp (sub (32, 52, read-sp (s)), mc-mem (s), 68)
  $\land$ disjoint (a, $4 * n$, sub (32, 52, read-sp (s)), 68))

THEOREM: qsort-statep-sp
qsort-statep $(s,\ a,\ l,\ r,\ n,\ lst) \rightarrow$ qsort-sp $(s,\ a,\ l,\ r,\ n,\ lst)$

```
; the conditions of the intermediate state s0.  s0 is the state of the
; machine right before the for statement in the corresponding C program.
```

DEFINITION:
qsort-s0p $(s,\ s0,\ a,\ l,\ r,\ n,\ lst,\ last,\ i)$
$=$ ((linked-rts-addr $(s0)$ = rts-addr (s))
  $\land$ (linked-a6 $(s0)$ = read-an (32, 6, s))
  $\land$ equal* (read-rn (32, 14, mc-rfile $(s0)$), sub (32, 4, read-sp (s)))
  $\land$ equal* (read-rn (32, 15, mc-rfile $(s0)$), sub (32, 24, read-sp (s)))
  $\land$ (movem-saved ($s0$, 4, 20, 5)
    = readm-rn (32, `'(2 3 4 10 11)`, mc-rfile (s)))
  $\land$ (mc-status $(s0)$ = `'running`)
  $\land$ evenp (mc-pc $(s0)$)
  $\land$ rom-addrp (sub (32, 82, mc-pc $(s0)$), mc-mem $(s0)$, 138)
  $\land$ mcode-addrp (sub (32, 82, mc-pc $(s0)$), mc-mem $(s0)$, QSORT-CODE)
  $\land$ ram-addrp (sub (32, 52, read-sp (s)), mc-mem $(s0)$, 68)
  $\land$ ram-addrp (a, mc-mem $(s0)$, $4 * n$)
  $\land$ mem-ilst (4, a, mc-mem $(s0)$, n, lst)
  $\land$ disjoint (a, $4 * n$, sub (32, 52, read-sp (s)), 68)
  $\land$ (a = read-an (32, 3, $s0$))
  $\land$ (l = nat-to-int (read-rn (32, 3, mc-rfile $(s0)$), 32))
  $\land$ (r = nat-to-int (read-rn (32, 4, mc-rfile $(s0)$), 32))

$\wedge \quad (\mathit{last} = \text{nat-to-int}\,(\text{read-rn}\,(\mathtt{32},\ \mathtt{2},\ \text{mc-rfile}\,(\mathit{s0})),\ \mathtt{32}))$
$\wedge \quad (i = \text{nat-to-int}\,(\text{read-rn}\,(\mathtt{32},\ \mathtt{0},\ \text{mc-rfile}\,(\mathit{s0})),\ \mathtt{32}))$
$\wedge \quad (l \in \mathbf{N})$
$\wedge \quad (r \in \mathbf{N})$
$\wedge \quad (n \in \mathbf{N})$
$\wedge \quad (\mathit{last} \in \mathbf{N})$
$\wedge \quad (i \in \mathbf{N})$
$\wedge \quad (l < r)$
$\wedge \quad (\mathit{last} < i)$
$\wedge \quad (i \leq (1 + r))$
$\wedge \quad (r < n)$
$\wedge \quad \text{int-rangep}\,(n,\ \mathtt{32})$
$\wedge \quad \text{uint-rangep}\,(\mathtt{4} * n,\ \mathtt{32}))$

```
; the conditions of the intermediate state s1.  s1 is the machine
; state after the execution of the first JSR instruction, but before
; the recursive execution of QSORT.
```

DEFINITION:
$\text{qsort-s1p}\,(s,\ \mathit{s1},\ a,\ l,\ r,\ n,\ \mathit{lst},\ \mathit{last})$
$= \quad ((\text{linked-rts-addr}\,(\mathit{s1}) = \text{rts-addr}\,(s))$
$\wedge \quad (\text{linked-a6}\,(\mathit{s1}) = \text{read-an}\,(\mathtt{32},\ \mathtt{6},\ s))$
$\wedge \quad \text{equal*}\,(\text{read-rn}\,(\mathtt{32},\ \mathtt{14},\ \text{mc-rfile}\,(\mathit{s1})),\ \text{sub}\,(\mathtt{32},\ \mathtt{4},\ \text{read-sp}\,(s)))$
$\wedge \quad \text{equal*}\,(\text{read-rn}\,(\mathtt{32},\ \mathtt{15},\ \text{mc-rfile}\,(\mathit{s1})),\ \text{sub}\,(\mathtt{32},\ \mathtt{40},\ \text{read-sp}\,(s)))$
$\wedge \quad (\text{movem-saved}\,(\mathit{s1},\ \mathtt{4},\ \mathtt{20},\ \mathtt{5})$
$\qquad = \quad \text{readm-rn}\,(\mathtt{32},\ \mathtt{'(2\ 3\ 4\ 10\ 11)},\ \text{mc-rfile}\,(s)))$
$\wedge \quad (\text{mc-status}\,(\mathit{s1}) = \mathtt{'running})$
$\wedge \quad (\text{mc-pc}\,(\mathit{s1}) = \text{read-an}\,(\mathtt{32},\ \mathtt{2},\ \mathit{s1}))$
$\wedge \quad \text{evenp}\,(\text{read-an}\,(\mathtt{32},\ \mathtt{2},\ \mathit{s1}))$
$\wedge \quad \text{evenp}\,(\text{rts-addr}\,(\mathit{s1}))$
$\wedge \quad \text{rom-addrp}\,(\text{read-an}\,(\mathtt{32},\ \mathtt{2},\ \mathit{s1}),\ \text{mc-mem}\,(\mathit{s1}),\ \mathtt{138})$
$\wedge \quad \text{mcode-addrp}\,(\text{read-an}\,(\mathtt{32},\ \mathtt{2},\ \mathit{s1}),\ \text{mc-mem}\,(\mathit{s1}),\ \textsc{qsort-code})$
$\wedge \quad \text{rom-addrp}\,(\text{sub}\,(\mathtt{32},\ \mathtt{116},\ \text{rts-addr}\,(\mathit{s1})),\ \text{mc-mem}\,(\mathit{s1}),\ \mathtt{138})$
$\wedge \quad \text{mcode-addrp}\,(\text{sub}\,(\mathtt{32},\ \mathtt{116},\ \text{rts-addr}\,(\mathit{s1})),\ \text{mc-mem}\,(\mathit{s1}),\ \textsc{qsort-code})$
$\wedge \quad \text{ram-addrp}\,(\text{sub}\,(\mathtt{32},\ \mathtt{52},\ \text{read-sp}\,(s)),\ \text{mc-mem}\,(\mathit{s1}),\ \mathtt{68})$
$\wedge \quad \text{ram-addrp}\,(a,\ \text{mc-mem}\,(\mathit{s1}),\ \mathtt{4} * n)$
$\wedge \quad \text{mem-ilst}\,(\mathtt{4},\ a,\ \text{mc-mem}\,(\mathit{s1}),\ n,\ \mathit{lst})$
$\wedge \quad \text{disjoint}\,(a,\ \mathtt{4} * n,\ \text{sub}\,(\mathtt{32},\ \mathtt{52},\ \text{read-sp}\,(s)),\ \mathtt{68})$
$\wedge \quad (a = \text{read-an}\,(\mathtt{32},\ \mathtt{3},\ \mathit{s1}))$
$\wedge \quad (l = \text{nat-to-int}\,(\text{read-rn}\,(\mathtt{32},\ \mathtt{3},\ \text{mc-rfile}\,(\mathit{s1})),\ \mathtt{32}))$
$\wedge \quad (r = \text{nat-to-int}\,(\text{read-rn}\,(\mathtt{32},\ \mathtt{4},\ \text{mc-rfile}\,(\mathit{s1})),\ \mathtt{32}))$
$\wedge \quad (\mathit{last} = \text{nat-to-int}\,(\text{read-rn}\,(\mathtt{32},\ \mathtt{2},\ \text{mc-rfile}\,(\mathit{s1})),\ \mathtt{32}))$
$\wedge \quad (a = \text{read-mem}\,(\text{add}\,(\mathtt{32},\ \text{read-sp}\,(\mathit{s1}),\ \mathtt{4}),\ \text{mc-mem}\,(\mathit{s1}),\ \mathtt{4}))$
$\wedge \quad (l = \text{iread-mem}\,(\text{add}\,(\mathtt{32},\ \text{read-sp}\,(\mathit{s1}),\ \mathtt{8}),\ \text{mc-mem}\,(\mathit{s1}),\ \mathtt{4}))$

$\wedge$ (iread-mem (add (32, read-sp ($s1$), 12), mc-mem ($s1$), 4)
  $=$ idifference ($last$, 1))
$\wedge$ ($l \in \mathbf{N}$)
$\wedge$ ($r \in \mathbf{N}$)
$\wedge$ ($last \in \mathbf{N}$)
$\wedge$ ($l < r$)
$\wedge$ ($last \leq r$)
$\wedge$ ($r < n$)
$\wedge$ int-rangep (n, 32)
$\wedge$ uint-rangep ($4 * n$, 32))

```
; the conditions of the intermediate state s2.  s2 is the machine
; state right after the first recursive call to QSORT.
```

DEFINITION:
qsort-s2p (s, $s2$, a, l, r, n, lst, $last$)
$=$ ((linked-rts-addr ($s2$) $=$ rts-addr (s))
$\wedge$ (linked-a6 ($s2$) $=$ read-an (32, 6, s))
$\wedge$ equal* (read-rn (32, 14, mc-rfile ($s2$)), sub (32, 4, read-sp (s)))
$\wedge$ equal* (read-rn (32, 15, mc-rfile ($s2$)), sub (32, 36, read-sp (s)))
$\wedge$ (movem-saved ($s2$, 4, 20, 5)
  $=$ readm-rn (32, '(2 3 4 10 11), mc-rfile (s)))
$\wedge$ (mc-status ($s2$) $=$ 'running)
$\wedge$ evenp (mc-pc ($s2$))
$\wedge$ evenp (read-an (32, 2, $s2$))
$\wedge$ rom-addrp (read-an (32, 2, $s2$), mc-mem ($s2$), 138)
$\wedge$ mcode-addrp (read-an (32, 2, $s2$), mc-mem ($s2$), QSORT-CODE)
$\wedge$ rom-addrp (sub (32, 116, mc-pc ($s2$)), mc-mem ($s2$), 138)
$\wedge$ mcode-addrp (sub (32, 116, mc-pc ($s2$)), mc-mem ($s2$), QSORT-CODE)
$\wedge$ ram-addrp (sub (32, 16, read-sp ($s2$)), mc-mem ($s2$), 68)
$\wedge$ ram-addrp (a, mc-mem ($s2$), $4 * n$)
$\wedge$ mem-ilst (4, a, mc-mem ($s2$), n, lst)
$\wedge$ disjoint (a, $4 * n$, sub (32, 16, read-sp ($s2$)), 68)
$\wedge$ ($a =$ read-an (32, 3, $s2$))
$\wedge$ ($l =$ nat-to-int (read-rn (32, 3, mc-rfile ($s2$)), 32))
$\wedge$ ($r =$ nat-to-int (read-rn (32, 4, mc-rfile ($s2$)), 32))
$\wedge$ ($last =$ nat-to-int (read-rn (32, 2, mc-rfile ($s2$)), 32))
$\wedge$ ($l \in \mathbf{N}$)
$\wedge$ ($r \in \mathbf{N}$)
$\wedge$ ($last \in \mathbf{N}$)
$\wedge$ ($l < r$)
$\wedge$ ($last \leq r$)
$\wedge$ ($r < n$)
$\wedge$ int-rangep (n, 32)
$\wedge$ uint-rangep ($4 * n$, 32))

```
; the conditions of the intermediate state s3.  s3 is the machine
; state right after the execution of the second JSR instruction, but
; before the execution of QSORT.
```

DEFINITION:
qsort-s3p (s, $s3$, a, l, r, n, lst, $last$)
$=$ ((linked-rts-addr ($s3$) = rts-addr (s))
$\wedge$ (linked-a6 ($s3$) = read-an (32, 6, s))
$\wedge$ equal* (read-rn (32, 14, mc-rfile ($s3$)), sub (32, 4, read-sp (s)))
$\wedge$ equal* (read-rn (32, 15, mc-rfile ($s3$)), sub (32, 52, read-sp (s)))
$\wedge$ (movem-saved ($s3$, 4, 20, 5)
$=$ readm-rn (32, '(2 3 4 10 11), mc-rfile (s)))
$\wedge$ (mc-status ($s3$) = 'running)
$\wedge$ (mc-pc ($s3$) = read-an (32, 2, $s3$))
$\wedge$ evenp (read-an (32, 2, $s3$))
$\wedge$ evenp (rts-addr ($s3$))
$\wedge$ rom-addrp (read-an (32, 2, $s3$), mc-mem ($s3$), 138)
$\wedge$ mcode-addrp (read-an (32, 2, $s3$), mc-mem ($s3$), QSORT-CODE)
$\wedge$ rom-addrp (sub (32, 128, rts-addr ($s3$)), mc-mem ($s3$), 138)
$\wedge$ mcode-addrp (sub (32, 128, rts-addr ($s3$)), mc-mem ($s3$), QSORT-CODE)
$\wedge$ ram-addrp (read-sp ($s3$), mc-mem ($s3$), 68)
$\wedge$ ram-addrp (a, mc-mem ($s3$), $4 * n$)
$\wedge$ mem-ilst (4, a, mc-mem ($s3$), n, lst)
$\wedge$ disjoint (a, $4 * n$, read-sp ($s3$), 68)
$\wedge$ (a = read-an (32, 3, $s3$))
$\wedge$ (l = nat-to-int (read-rn (32, 3, mc-rfile ($s3$)), 32))
$\wedge$ (r = nat-to-int (read-rn (32, 4, mc-rfile ($s3$)), 32))
$\wedge$ ($last$ = nat-to-int (read-rn (32, 2, mc-rfile ($s3$)), 32))
$\wedge$ (a = read-mem (add (32, read-sp ($s3$), 4), mc-mem ($s3$), 4))
$\wedge$ (iread-mem (add (32, read-sp ($s3$), 8), mc-mem ($s3$), 4)
$=$ ($1 + last$))
$\wedge$ (r = iread-mem (add (32, read-sp ($s3$), 12), mc-mem ($s3$), 4))
$\wedge$ ($l \in \mathbf{N}$)
$\wedge$ ($r \in \mathbf{N}$)
$\wedge$ ($last \in \mathbf{N}$)
$\wedge$ ($l < r$)
$\wedge$ ($last \leq r$)
$\wedge$ ($r < n$)
$\wedge$ int-rangep (n, 32)
$\wedge$ uint-rangep ($4 * n$, 32))

```
; the conditions of the intermediate state s4.  s4 is the machine
; state right after the second recursive call to QSORT.
```

DEFINITION:

qsort-s4p $(s,\ s4,\ a,\ l,\ r,\ n,\ lst)$
$=$ ((linked-rts-addr $(s4)$ = rts-addr (s))
∧ (linked-a6 $(s4)$ = read-an (32, 6, s))
∧ equal* (read-rn (32, 14, mc-rfile $(s4)$), sub (32, 4, read-sp (s)))
∧ equal* (read-rn (32, 15, mc-rfile $(s4)$), sub (32, 48, read-sp (s)))
∧ (movem-saved $(s4$, 4, 20, 5)
= readm-rn (32, `'(2 3 4 10 11)`, mc-rfile (s)))
∧ (mc-status $(s4)$ = `'running`)
∧ evenp (mc-pc $(s4)$)
∧ rom-addrp (sub (32, 128, mc-pc $(s4)$), mc-mem $(s4)$, 138)
∧ mcode-addrp (sub (32, 128, mc-pc $(s4)$), mc-mem $(s4)$, QSORT-CODE)
∧ ram-addrp (sub (32, 48, read-an (32, 6, $s4$)), mc-mem $(s4)$, 68)
∧ ram-addrp (a, mc-mem $(s4)$, $4 * n$)
∧ mem-ilst (4, a, mc-mem $(s4)$, n, lst)
∧ disjoint (a, $4 * n$, sub (32, 48, read-an (32, 6, $s4$)), 68))

```
; the conditions of the final state.  s5 is the machine state after
; the execution of this QSORT program.
```

DEFINITION:
qsort-s5p $(s,\ s5,\ a,\ l,\ r,\ n,\ lst)$
$=$ ((mc-status $(s5)$ = `'running`)
∧ (mc-pc $(s5)$ = rts-addr (s))
∧ (read-rn (32, 14, mc-rfile $(s5)$)
= read-rn (32, 14, mc-rfile (s))))
∧ (read-rn (32, 15, mc-rfile $(s5)$) = add (32, read-sp (s), 4))
∧ mem-ilst (4, a, mc-mem $(s5)$, n, lst)
∧ (read-dn (32, 2, $s5$) = read-dn (32, 2, s))
∧ (read-dn (32, 3, $s5$) = read-dn (32, 3, s))
∧ (read-dn (32, 4, $s5$) = read-dn (32, 4, s))
∧ (read-an (32, 2, $s5$) = read-an (32, 2, s))
∧ (read-an (32, 3, $s5$) = read-an (32, 3, s)))

DEFINITION:
qsort-sk $(s,\ s5,\ a,\ l,\ r,\ n,\ lst)$
$\leftrightarrow$ $\forall\ x,\ k$ **let** sp **be** sub (32, qstack $(l,\ r,\ lst)$ − 16, read-sp (s))
in
(disjoint (sp, qstack $(l,\ r,\ lst)$, x, k)
∧ disjoint (a, $4 * n$, x, k))
→ (read-mem (x, mc-mem $(s5)$, k)
= read-mem (x, mc-mem (s), k)) **endlet**

EVENT: Disable qsort-sk.

THEOREM: qsort-sk-1

let sp **be** sub (32, qstack (l, r, lst) − 16, read-sp (s))
in
(qsort-sk (s, $s5$, a, l, r, n, lst)
 ∧ disjoint (sp, qstack (l, r, lst), x, k)
 ∧ disjoint (a, 4 ∗ n, x, k))
→ (read-mem (x, mc-mem ($s5$), k) = read-mem (x, mc-mem (s), k)) **endlet**

THEOREM: qsort-sk-2
let sp **be** sub (32, qstack (l, r, lst) − 16, read-sp (s))
in
(qsort-sk (s, $s5$, a, l, r, n, lst)
 ∧ disjoint (sp, qstack (l, r, lst), x, $opsz$ ∗ k)
 ∧ disjoint (a, 4 ∗ n, x, $opsz$ ∗ k))
→ (readm-mem ($opsz$, x, mc-mem ($s5$), k)
 = readm-mem ($opsz$, x, mc-mem (s), k)) **endlet**

EVENT: Disable qsort-sk-1.

EVENT: Disable qsort-sk-2.

```
; base case: from the initial state to exit (s --> exit).
```

THEOREM: qsort-base
(qsort-sp (s, a, l, r, n, lst) ∧ ($l \not< r$))
→ qsort-s5p (s, stepn (s, qsort-10 (a, l, r, n, lst)), a, l, r, n, lst)

THEOREM: qsort-base-rfile
(qsort-sp (s, a, l, r, n, lst) ∧ ($l \not< r$) ∧ d2-7a2-5p (rn) ∧ ($oplen \leq 32$))
→ (read-rn ($oplen$, rn, mc-rfile (stepn (s, qsort-10 (a, l, r, n, lst))))
 = read-rn ($oplen$, rn, mc-rfile (s)))

THEOREM: qsort-base-mem
(qsort-sp (s, a, l, r, n, lst)
 ∧ ($l \not< r$)
 ∧ disjoint (sub (32, 24, read-sp (s)), 40, x, k))
→ (read-mem (x, mc-mem (stepn (s, 10)), k) = read-mem (x, mc-mem (s), k))

THEOREM: qsort-base-mem-sk
(qsort-sp (s, a, l, r, n, lst) ∧ ($l \not< r$))
→ qsort-sk (s, stepn (s, qsort-10 (a, l, r, n, lst)), a, l, r, n, lst)

```
; induction case: s --> s0 --> s1 --> s2 --> s3 --> s4 --> exit.
; s --> s0:
```

THEOREM: add1-int-rangep
$(x < \text{nat-to-int}\,(y,\, n)) \rightarrow \text{int-rangep}\,(1 + x,\, n)$

THEOREM: mean-bounds
$(i < j) \rightarrow ((((i + j) \div 2) < j) \wedge (((i + j) \div 2) \not< i))$

THEOREM: int-rangep-plus-1
$(\text{int-rangep}\,(2 * j,\, n) \wedge (i < j)) \rightarrow \text{int-rangep}\,(i + j,\, n)$

THEOREM: qsort-s-s0
let *lst1* **be** swap$\,(l,\, (l + r) \div 2,\, lst)$
in
$(\text{qsort-sp}\,(s,\, a,\, l,\, r,\, n,\, lst) \wedge (l < r))$
$\rightarrow\quad \text{qsort-s0p}\,(s,\, \text{stepn}\,(s,\, 18),\, a,\, l,\, r,\, n,\, lst1,\, l,\, 1 + l)$ **endlet**

THEOREM: qsort-s-s0-rfile
$(\text{qsort-sp}\,(s,\, a,\, l,\, r,\, n,\, lst) \wedge (l < r) \wedge \text{d5-7a4-5p}\,(rn))$
$\rightarrow\quad (\text{read-rn}\,(oplen,\, rn,\, \text{mc-rfile}\,(\text{stepn}\,(s,\, 18)))$
$\qquad = \quad \text{read-rn}\,(oplen,\, rn,\, \text{mc-rfile}\,(s)))$

THEOREM: qsort-s-s0-mem
$(\text{qsort-sp}\,(s,\, a,\, l,\, r,\, n,\, lst)$
$\wedge\quad (l < r)$
$\wedge\quad \text{disjoint}\,(\text{sub}\,(32,\, 52,\, \text{read-sp}\,(s)),\, 68,\, x,\, k)$
$\wedge\quad \text{disjoint}\,(a,\, 4 * n,\, x,\, k))$
$\rightarrow\quad (\text{read-mem}\,(x,\, \text{mc-mem}\,(\text{stepn}\,(s,\, 18)),\, k) = \text{read-mem}\,(x,\, \text{mc-mem}\,(s),\, k))$

```
; s0 --> s1:
; base case (s0 --> s1):
```

THEOREM: qsort-s0-s1
$(\text{qsort-s0p}\,(s,\, s0,\, a,\, l,\, r,\, n,\, lst,\, last,\, i) \wedge (r < i) \wedge (last1 = \text{fix}\,(last)))$
$\rightarrow\quad \text{qsort-s1p}\,(s,\, \text{stepn}\,(s0,\, 11),\, a,\, l,\, r,\, n,\, \text{swap}\,(l,\, last,\, lst),\, last1)$

THEOREM: qsort-s0-s1-rfile
$(\text{qsort-s0p}\,(s,\, s0,\, a,\, l,\, r,\, n,\, lst,\, last,\, i) \wedge (r < i) \wedge \text{d5-7a4-5p}\,(rn))$
$\rightarrow\quad (\text{read-rn}\,(oplen,\, rn,\, \text{mc-rfile}\,(\text{stepn}\,(s0,\, 11)))$
$\qquad = \quad \text{read-rn}\,(oplen,\, rn,\, \text{mc-rfile}\,(s0)))$

THEOREM: qsort-s0-s1-mem
$(\text{qsort-s0p}\,(s,\, s0,\, a,\, l,\, r,\, n,\, lst,\, last,\, i)$
$\wedge\quad (r < i)$
$\wedge\quad \text{disjoint}\,(\text{sub}\,(32,\, 52,\, \text{read-sp}\,(s)),\, 68,\, x,\, k)$
$\wedge\quad \text{disjoint}\,(a,\, 4 * n,\, x,\, k))$
$\rightarrow\quad (\text{read-mem}\,(x,\, \text{mc-mem}\,(\text{stepn}\,(s0,\, 11)),\, k) = \text{read-mem}\,(x,\, \text{mc-mem}\,(s0),\, k))$

`; induction case (s0 --> s0):`

THEOREM: add1-int-rangepxx
$((i \leq r) \wedge (r < n) \wedge \text{int-rangep}\,(n,\ \texttt{32})) \rightarrow \text{int-rangep}\,(1 + i,\ \texttt{32})$

THEOREM: qsort-s0-s0-1
$(\text{qsort-s0p}\,(s,\ s0,\ a,\ l,\ r,\ n,\ lst,\ last,\ i)$
$\wedge \quad (r \not< i)$
$\wedge \quad (\neg\ \text{ilessp}\,(\text{get-nth}\,(i,\ lst),\ \text{get-nth}\,(l,\ lst))))$
$\rightarrow \quad \text{qsort-s0p}\,(s,\ \text{stepn}\,(s0,\ \texttt{6}),\ a,\ l,\ r,\ n,\ lst,\ last,\ 1 + i)$

THEOREM: qsort-s0-s0-2
$(\text{qsort-s0p}\,(s,\ s0,\ a,\ l,\ r,\ n,\ lst,\ last,\ i)$
$\wedge \quad (r \not< i)$
$\wedge \quad \text{ilessp}\,(\text{get-nth}\,(i,\ lst),\ \text{get-nth}\,(l,\ lst)))$
$\rightarrow \quad \text{qsort-s0p}\,(s,\ \text{stepn}\,(s0,\ \texttt{10}),\ a,\ l,\ r,\ n,\ \text{swap}\,(1 + last,\ i,\ lst),\ 1 + last,\ 1 + i)$

THEOREM: qsort-s0-s0-rfile-1
$(\text{qsort-s0p}\,(s,\ s0,\ a,\ l,\ r,\ n,\ lst,\ last,\ i)$
$\wedge \quad (r \not< i)$
$\wedge \quad (\neg\ \text{ilessp}\,(\text{get-nth}\,(i,\ lst),\ \text{get-nth}\,(l,\ lst)))$
$\wedge \quad \text{d5-7a4-5p}\,(rn))$
$\rightarrow \quad (\text{read-rn}\,(oplen,\ rn,\ \text{mc-rfile}\,(\text{stepn}\,(s0,\ \texttt{6})))$
$\qquad = \quad \text{read-rn}\,(oplen,\ rn,\ \text{mc-rfile}\,(s0)))$

THEOREM: qsort-s0-s0-rfile-2
$(\text{qsort-s0p}\,(s,\ s0,\ a,\ l,\ r,\ n,\ lst,\ last,\ i)$
$\wedge \quad (r \not< i)$
$\wedge \quad \text{ilessp}\,(\text{get-nth}\,(i,\ lst),\ \text{get-nth}\,(l,\ lst))$
$\wedge \quad \text{d5-7a4-5p}\,(rn))$
$\rightarrow \quad (\text{read-rn}\,(oplen,\ rn,\ \text{mc-rfile}\,(\text{stepn}\,(s0,\ \texttt{10})))$
$\qquad = \quad \text{read-rn}\,(oplen,\ rn,\ \text{mc-rfile}\,(s0)))$

THEOREM: qsort-s0-s0-mem-1
$(\text{qsort-s0p}\,(s,\ s0,\ a,\ l,\ r,\ n,\ lst,\ last,\ i)$
$\wedge \quad (r \not< i)$
$\wedge \quad (\neg\ \text{ilessp}\,(\text{get-nth}\,(i,\ lst),\ \text{get-nth}\,(l,\ lst))))$
$\rightarrow \quad (\text{read-mem}\,(x,\ \text{mc-mem}\,(\text{stepn}\,(s0,\ \texttt{6})),\ k) = \text{read-mem}\,(x,\ \text{mc-mem}\,(s0),\ k))$

THEOREM: qsort-s0-s0-mem-2
$(\text{qsort-s0p}\,(s,\ s0,\ a,\ l,\ r,\ n,\ lst,\ last,\ i)$
$\wedge \quad (r \not< i)$
$\wedge \quad \text{ilessp}\,(\text{get-nth}\,(i,\ lst),\ \text{get-nth}\,(l,\ lst))$
$\wedge \quad \text{disjoint}\,(a,\ \texttt{4} * n,\ x,\ k))$
$\rightarrow \quad (\text{read-mem}\,(x,\ \text{mc-mem}\,(\text{stepn}\,(s0,\ \texttt{10})),\ k) = \text{read-mem}\,(x,\ \text{mc-mem}\,(s0),\ k))$

```
; induction hint for the partition.
```

DEFINITION:
qpart-induct (s, l, r, *lst*, *last*, i)
$=$ **if** $r < i$ **then t**
elseif ilessp (get-nth (i, *lst*), get-nth (l, *lst*))
then qpart-induct (stepn (s, `10`),
l,
r,
swap ($1 +$ *last*, i, *lst*),
$1 +$ *last*,
$1 + i$)
else qpart-induct (stepn (s, `6`), l, r, *lst*, *last*, $1 + i$) **endif**

THEOREM: qpart-aux-ct
qsort-s0p (s, *s0*, a, l, r, n, *lst*, *last*, i)
$\rightarrow$ qsort-s1p (s,
stepn (*s0*, qpart-aux-t (a, l, r, n, *lst*, *last*, i)),
a,
l,
r,
n,
qpart-aux (l, r, *lst*, *last*, i),
qlast-aux (l, r, *lst*, *last*, i))

THEOREM: qsort-s-s1
(qsort-sp (s, a, l, r, n, *lst*) $\wedge$ ($l < r$))
$\rightarrow$ qsort-s1p (s,
stepn (s, qpart-t (a, l, r, n, *lst*)),
a,
l,
r,
n,
qpart (l, r, *lst*),
qlast (l, r, *lst*))

THEOREM: qpart-aux-rfile
let *s1* **be** stepn (*s0*, qpart-aux-t (a, l, r, n, *lst*, *last*, i))
in
(qsort-s0p (s, *s0*, a, l, r, n, *lst*, *last*, i) $\wedge$ d5-7a4-5p (*rn*))
$\rightarrow$ (read-rn (*oplen*, *rn*, mc-rfile (*s1*))
$=$ read-rn (*oplen*, *rn*, mc-rfile (*s0*))) **endlet**

THEOREM: qsort-s-s1-rfile
(qsort-sp (s, a, l, r, n, *lst*) $\wedge$ ($l < r$) $\wedge$ d5-7a4-5p (*rn*))

$\rightarrow$ (read-rn ($oplen$, rn, mc-rfile (stepn (s, qpart-t (a, l, r, n, lst))))
= read-rn ($oplen$, rn, mc-rfile (s)))

THEOREM: qpart-aux-mem
let $s1$ **be** stepn ($s0$, qpart-aux-t (a, l, r, n, lst, $last$, i))
in
(qsort-s0p (s, $s0$, a, l, r, n, lst, $last$, i)
$\wedge$ disjoint (sub (32, 52, read-sp (s)), 68, x, k)
$\wedge$ disjoint (a, $4 * n$, x, k))
$\rightarrow$ (read-mem (x, mc-mem ($s1$), k) = read-mem (x, mc-mem ($s0$), k)) **endlet**

THEOREM: qsort-sk-s-s1
(qsort-sp (s, a, l, r, n, lst) $\wedge$ ($l < r$))
$\rightarrow$ qsort-sk (s, stepn (s, qpart-t (a, l, r, n, lst)), a, l, r, n, lst)

EVENT: Disable qpart-t.

```
; s2 --> s3:
```

THEOREM: qsort-s2-s3
let $qlst$ **be** qpart (l, r, lst),
$last$ **be** qlast (l, r, lst)
in
qsort-s2p (s, $s2$, a, l, r, n, qsort (l, $last - 1$, $qlst$), $last$)
$\rightarrow$ qsort-s3p (s,
stepn ($s2$, qsort-5 (a, l, r, n, lst)),
a,
l,
r,
n,
qsort (l, $last - 1$, $qlst$),
$last$) **endlet**

THEOREM: qsort-s2-s3-rfile-la
(qsort-s2p (s, $s2$, a, l, r, n, $lst1$, $last$) $\wedge$ d5-7a4-5p (rn))
$\rightarrow$ (read-rn ($oplen$, rn, mc-rfile (stepn ($s2$, 5)))
= read-rn ($oplen$, rn, mc-rfile ($s2$)))

THEOREM: qsort-s2-s3-rfile
let $qlst$ **be** qpart (l, r, lst),
$last$ **be** qlast (l, r, lst),
$s2$ **be** stepn (stepn (s, $k1$), $k2$)
in
(qsort-s2p (s, $s2$, a, l, r, n, qsort (l, $last - 1$, $qlst$), $last$)
$\wedge$ d5-7a4-5p (rn))

$\rightarrow$ (read-rn (*oplen*, *rn*, mc-rfile (stepn (*s2*, qsort-5 (*a*, *l*, *r*, *n*, *lst*))))
= read-rn (*oplen*, *rn*, mc-rfile (*s2*))) **endlet**

THEOREM: qsort-s2-s3-mem
(qsort-s2p (*s*, *s2*, *a*, *l*, *r*, *n*, *lst*, *last*)
$\wedge$ disjoint (sub (`32`, `52`, read-sp (*s*)), `68`, *x*, *k*))
$\rightarrow$ (read-mem (*x*, mc-mem (stepn (*s2*, `5`)), *k*) = read-mem (*x*, mc-mem (*s2*), *k*))

THEOREM: qsort-sk-s-s3
let *lst1* **be** qpart (*l*, *r*, *lst*),
last **be** qlast (*l*, *r*, *lst*)
in
(qsort-s2p (*s*, *s2*, *a*, *l*, *r*, *n*, qsort (*l*, *last* − 1, *lst1*), *last*)
$\wedge$ qsort-sk (*s*, *s2*, *a*, *l*, *r*, *n*, *lst*))
$\rightarrow$ qsort-sk (*s*, stepn (*s2*, qsort-5 (*a*, *l*, *r*, *n*, *lst*)), *a*, *l*, *r*, *n*, *lst*) **endlet**

```
; s4 --> exit:
```

THEOREM: qsort-s4-s5
let *qlst* **be** qpart (*l*, *r*, *lst*),
last **be** qlast (*l*, *r*, *lst*)
in
qsort-s4p (*s*, *s4*, *a*, *l*, *r*, *n*, qsort (1 + *last*, *r*, qsort (*l*, *last* − 1, *qlst*)))
$\rightarrow$ qsort-s5p (*s*,
stepn (*s4*, qsort-3 (*a*, *l*, *r*, *n*, *lst*)),
a,
l,
r,
n,
qsort (1 + *last*, *r*, qsort (*l*, *last* − 1, *qlst*))) **endlet**

THEOREM: qsort-s4-s5-rfile-la
(qsort-s4p (*s*, *s4*, *a*, *l*, *r*, *n*, qsort (*l*, *r*, *lst*))
$\wedge$ (*oplen* ≤ `32`)
$\wedge$ d2-7a2-5p (*rn*))
$\rightarrow$ (read-rn (*oplen*, *rn*, mc-rfile (stepn (*s4*, `3`)))
= **if** d5-7a4-5p (*rn*) **then** read-rn (*oplen*, *rn*, mc-rfile (*s4*))
else read-rn (*oplen*, *rn*, mc-rfile (*s*)) **endif**)

THEOREM: qsort-s4-s5-rfile
let *s4* **be** stepn (stepn (stepn (stepn (*s*, *k1*), *k2*), *k3*), *k4*)
in
(qsort-s4p (*s*, *s4*, *a*, *l*, *r*, *n*, qsort (*l*, *r*, *lst*))
$\wedge$ (*oplen* ≤ `32`)
$\wedge$ d2-7a2-5p (*rn*))

$\rightarrow \quad (\text{read-rn}\,(\mathit{oplen},\ \mathit{rn},\ \text{mc-rfile}\,(\text{stepn}\,(\mathit{s4},\ \text{qsort-3}\,(a,\ l,\ r,\ n,\ \mathit{lst})))))$
$\quad = \quad \textbf{if}\ \text{d5-7a4-5p}\,(\mathit{rn})\ \textbf{then}\ \text{read-rn}\,(\mathit{oplen},\ \mathit{rn},\ \text{mc-rfile}\,(\mathit{s4}))$
$\quad\quad \textbf{else}\ \text{read-rn}\,(\mathit{oplen},\ \mathit{rn},\ \text{mc-rfile}\,(s))\ \textbf{endif})\ \textbf{endlet}$

THEOREM: qsort-s4-s5-mem
$(\text{qsort-s4p}\,(s,\ \mathit{s4},\ a,\ l,\ r,\ n,\ \mathit{lst}) \land \text{disjoint}\,(\text{sub}\,(\texttt{32},\ \texttt{52},\ \text{read-sp}\,(s)),\ \texttt{68},\ x,\ k))$
$\rightarrow \quad (\text{read-mem}\,(x,\ \text{mc-mem}\,(\text{stepn}\,(\mathit{s4},\ \texttt{3})),\ k) = \text{read-mem}\,(x,\ \text{mc-mem}\,(\mathit{s4}),\ k))$

THEOREM: qsort-sk-s-s5
$(\text{qsort-s4p}\,(s,\ \mathit{s4},\ a,\ l,\ r,\ n,\ \text{qsort}\,(l,\ r,\ \mathit{lst})) \land \text{qsort-sk}\,(s,\ \mathit{s4},\ a,\ l,\ r,\ n,\ \mathit{lst}))$
$\rightarrow \quad \text{qsort-sk}\,(s,\ \text{stepn}\,(\mathit{s4},\ \text{qsort-3}\,(a,\ l,\ r,\ n,\ \mathit{lst})),\ a,\ l,\ r,\ n,\ \mathit{lst})$

```
; some auxiliary lemmas.
```

THEOREM: qlast-aux-0
$(r < i) \rightarrow (\text{qlast-aux}\,(l,\ r,\ \mathit{lst},\ \mathit{last},\ i) = \text{fix}\,(\mathit{last}))$

THEOREM: qlast-0
$(r \leq l) \rightarrow (\text{qlast}\,(l,\ r,\ \mathit{lst}) = \text{fix}\,(l))$

THEOREM: qstack-la3
$(l \not< r) \rightarrow (\text{qstack}\,(l,\ r,\ \mathit{lst}) = \texttt{68})$

THEOREM: qstack-0
$(\text{qstack}\,(l,\ \texttt{-1},\ \mathit{lst}) = \texttt{68})$
$\land \quad (\text{qstack}\,(l,\ \texttt{0},\ \mathit{lst}) = \texttt{68})$
$\land \quad (\text{qstack}\,(l,\ l,\ \mathit{lst}) = \texttt{68})$

THEOREM: qstack-la4
$\textbf{let}\ \mathit{last}\ \textbf{be}\ \text{qlast}\,(l,\ r,\ \mathit{lst}),$
$\quad \mathit{lst1}\ \textbf{be}\ \text{qpart}\,(l,\ r,\ \mathit{lst})$
$\textbf{in}$
$\text{qstack}\,(l,\ r,\ \mathit{lst}) \not< \text{qstack}\,(l,\ \mathit{last} - 1,\ \mathit{lst1})\ \textbf{endlet}$

```
; s1 --> s2:  the first recursive call.
```

THEOREM: qsort-s1-crock
$(((\mathit{q1} + \texttt{40}) \leq q) \land (\texttt{68} \leq \mathit{q1}) \land (q < \exp\,(\texttt{2},\ \texttt{32})))$
$\rightarrow \quad (q \not< (\mathit{q1} + \text{add}\,(\texttt{32},$
$\quad\quad \texttt{4294967256},$
$\quad\quad \text{add}\,(\texttt{32},\ \text{neg}\,(\texttt{32},\ \mathit{q1} - \texttt{16}),\ q - \texttt{16}))))$

THEOREM: qstack-la1~ $((\mathit{last} = \text{qlast}\,(l,\ r,\ \mathit{lst})) \land (\mathit{lst1} = \text{qpart}\,(l,\ r,\ \mathit{lst})) \land (l < r))$
$\rightarrow \quad (\text{qstack}\,(l,\ r,\ \mathit{lst}) \not< (\texttt{40} + \text{qstack}\,(l,\ \mathit{last} - 1,\ \mathit{lst1})))$

THEOREM: qsort-s1-s
let *lst1* **be** qpart $(l,\ r,\ lst)$,
last **be** qlast $(l,\ r,\ lst)$
in
$(\text{qsort-s1p}\,(s,\ \text{stepn}\,(s,\ k),\ a,\ l,\ r,\ n,\ lst1,\ last)$
$\land\quad \text{qsort-statep}\,(s,\ a,\ l,\ r,\ n,\ lst))$
$\to\quad \text{qsort-statep}\,(\text{stepn}\,(s,\ k),\ a,\ l,\ \text{idifference}\,(last,\ 1),\ n,\ lst1)$ **endlet**

THEOREM: qsort-s1-crock1
$((68 \leq q1)$
$\land\quad ((40 + q1) < 4294967296)$
$\land\quad ((x + k) \leq 4294967296)$
$\land\quad (4294967272 \leq x))$
$\to\quad \text{disjoint}\,(\text{add}\,(32,\ 4294967256,\ \text{neg}\,(32,\ q1 - 16)),\ q1,\ x,\ k)$

THEOREM: qsort-s1-crock2-crock
$((68 \leq q1) \land (q1 < 4294967256))$
$\to\quad (\text{add}\,(32,\ 4294967256,\ \text{neg}\,(32,\ q1 - 16)) \not< 16)$

THEOREM: qsort-s1-crock2
$((68 \leq q1) \land ((40 + q1) < 4294967296) \land ((x + k) \leq 16))$
$\to\quad \text{disjoint}\,(\text{add}\,(32,\ 4294967256,\ \text{neg}\,(32,\ q1 - 16)),\ q1,\ x,\ k)$

EVENT: Disable qsort-s1-crock2-crock.

THEOREM: qsort-s1-s2
let *lst1* **be** qpart $(l,\ r,\ lst)$,
last **be** qlast $(l,\ r,\ lst)$
in
$(\text{qsort-s1p}\,(s,\ s1,\ a,\ l,\ r,\ n,\ lst1,\ last)$
$\land\quad \text{qsort-statep}\,(s,\ a,\ l,\ r,\ n,\ lst)$
$\land\quad \text{qsort-s5p}\,(s1,$
$\text{stepn}\,(s1,\ k),$
$a,$
$l,$
$\text{idifference}\,(last,\ 1),$
$n,$
$\text{qsort}\,(l,\ last - 1,\ lst1))$
$\land\quad \text{qsort-sk}\,(s1,\ \text{stepn}\,(s1,\ k),\ a,\ l,\ \text{idifference}\,(last,\ 1),\ n,\ lst1))$
$\to\quad \text{qsort-s2p}\,(s,\ \text{stepn}\,(s1,\ k),\ a,\ l,\ r,\ n,\ \text{qsort}\,(l,\ last - 1,\ lst1),\ last)$ **endlet**

EVENT: Disable qsort-s1-crock1.

EVENT: Disable qsort-s1-crock2.

THEOREM: qsort-s1-s2-mem

$$
\begin{array}{l}
\textbf{let } \mathit{lst1} \textbf{ be } \mathrm{qpart}(l, r, \mathit{lst}), \\
\quad \mathit{last} \textbf{ be } \mathrm{qlast}(l, r, \mathit{lst}) \\
\textbf{in} \\
(\text{qsort-s1p}(s, \mathit{s1}, a, l, r, n, \mathit{lst1}, \mathit{last}) \\
\ \wedge \quad \text{qsort-statep}(s, a, l, r, n, \mathit{lst}) \\
\ \wedge \quad \text{qsort-sk}(s, \mathit{s1}, a, l, r, n, \mathit{lst}) \\
\ \wedge \quad \text{qsort-sk}(\mathit{s1}, \mathrm{stepn}(\mathit{s1}, k), a, l, \mathrm{idifference}(\mathit{last}, \mathtt{1}), n, \mathit{lst1}) \\
\ \wedge \quad \mathrm{disjoint}(a, \mathtt{4} * n, x, \mathit{k1}) \\
\ \wedge \quad \mathrm{disjoint}(\mathrm{sub}(\mathtt{32}, \mathrm{qstack}(l, r, \mathit{lst}) - \mathtt{16}, \text{read-sp}(s)), \\
\qquad\qquad\qquad \mathrm{qstack}(l, r, \mathit{lst}), \\
\qquad\qquad\qquad x, \\
\qquad\qquad\qquad \mathit{k1})) \\
\to \quad (\text{read-mem}(x, \text{mc-mem}(\mathrm{stepn}(\mathit{s1}, k)), \mathit{k1}) \\
\qquad = \quad \text{read-mem}(x, \text{mc-mem}(s), \mathit{k1})) \textbf{ endlet}
\end{array}
$$

EVENT: Disable qstack-la1~.

THEOREM: qsort-sk-s-s2

$$
\begin{array}{l}
\textbf{let } \mathit{lst1} \textbf{ be } \mathrm{qpart}(l, r, \mathit{lst}), \\
\quad \mathit{last} \textbf{ be } \mathrm{qlast}(l, r, \mathit{lst}) \\
\textbf{in} \\
(\text{qsort-s1p}(s, \mathit{s1}, a, l, r, n, \mathit{lst1}, \mathit{last}) \\
\ \wedge \quad \text{qsort-statep}(s, a, l, r, n, \mathit{lst}) \\
\ \wedge \quad \text{qsort-sk}(s, \mathit{s1}, a, l, r, n, \mathit{lst}) \\
\ \wedge \quad \text{qsort-sk}(\mathit{s1}, \mathrm{stepn}(\mathit{s1}, k), a, l, \mathrm{idifference}(\mathit{last}, \mathtt{1}), n, \mathit{lst1})) \\
\to \quad \text{qsort-sk}(s, \mathrm{stepn}(\mathit{s1}, k), a, l, r, n, \mathit{lst}) \textbf{ endlet}
\end{array}
$$

EVENT: Disable qsort-s1-s2-mem.

```
; s3 --> s4:  the second recursive call.
```

THEOREM: qsort-s3-crock

$$
\begin{array}{l}
(((\mathit{q1} + \mathtt{52}) \le q) \wedge (\mathtt{68} \le \mathit{q1}) \wedge (q < \exp(\mathtt{2}, \mathtt{32}))) \\
\to \quad (q \not< (\mathit{q1} + \mathrm{add}(\mathtt{32}, \\
\qquad\qquad\qquad\qquad \mathtt{4294967244}, \\
\qquad\qquad\qquad\qquad \mathrm{add}(\mathtt{32}, \mathrm{neg}(\mathtt{32}, \mathit{q1} - \mathtt{16}), q - \mathtt{16}))))
\end{array}
$$

THEOREM: qstack-la2~

$$
\begin{array}{l}
((\mathit{last} = \mathrm{qlast}(l, r, \mathit{lst})) \wedge (\mathit{lst1} = \mathrm{qpart}(l, r, \mathit{lst})) \wedge (l < r)) \\
\to \quad (\mathrm{qstack}(l, r, \mathit{lst}) \\
\qquad \not< \quad (\mathtt{52} + \mathrm{qstack}(1 + \mathit{last}, r, \mathrm{qsort}(l, \mathit{last} - 1, \mathit{lst1}))))
\end{array}
$$

THEOREM: qsort-s3-s-la

$\textbf{let } \mathit{lst1} \textbf{ be } \mathrm{qsort}(l, \mathrm{qlast}(l, r, \mathit{lst}) - 1, \mathrm{qpart}(l, r, \mathit{lst})),$

$\quad$ *last* **be** qlast (l, r, lst)
in
(qsort-s3p $(s, \text{stepn}(s, k), a, l, r, n, lst1, last)$
$\land$ qsort-statep $(s, a, l, r, n, lst))$
$\rightarrow$ qsort-statep $(\text{stepn}(s, k), a, 1 + last, r, n, lst1)$ **endlet**

THEOREM: qsort-s3-s
let *lst1* **be** qsort $(l, \text{qlast}(l, r, lst) - 1, \text{qpart}(l, r, lst))$,
$\quad$ *last* **be** qlast (l, r, lst)
in
(qsort-s3p $(s, \text{stepn}(\text{stepn}(\text{stepn}(s, k1), k2), k3), a, l, r, n, lst1, last)$
$\land$ qsort-statep $(s, a, l, r, n, lst))$
$\rightarrow$ qsort-statep (stepn (stepn (stepn $(s, k1), k2), k3)$,
a,
$1 + last$,
r,
n,
$lst1)$ **endlet**

THEOREM: qsort-s3-crock1
$((68 \leq q1)$
$\land$ $((52 + q1) < 4294967296)$
$\land$ $((x + k) \leq 4294967296)$
$\land$ $(4294967260 \leq x))$
$\rightarrow$ disjoint (add (32, 4294967244, neg (32, $q1 - 16$)), $q1, x, k$)

THEOREM: qsort-s3-crock2-crock
$((68 \leq q1) \land (q1 < 4294967244))$
$\rightarrow$ (add (32, 4294967244, neg (32, $q1 - 16$)) $\not< 16$)

THEOREM: qsort-s3-crock2
$((68 \leq q1) \land ((52 + q1) < 4294967296) \land ((x + k) \leq 16))$
$\rightarrow$ disjoint (add (32, 4294967244, neg (32, $q1 - 16$)), $q1, x, k$)

EVENT: Disable qsort-s3-crock2-crock.

THEOREM: qsort-s3-s4
let *lst1* **be** qsort $(l, \text{qlast}(l, r, lst) - 1, \text{qpart}(l, r, lst))$,
$\quad$ *last* **be** qlast (l, r, lst)
in
(qsort-s3p $(s, s3, a, l, r, n, lst1, last)$
$\land$ qsort-statep (s, a, l, r, n, lst)
$\land$ qsort-s5p $(s3$,
stepn $(s3, k)$,

a,
$1 + last$,
r,
n,
qsort $(1 + last,\ r,\ lst1))$
$\land$ qsort-sk $(s3,$ stepn $(s3,\ k),\ a,\ 1 + last,\ r,\ n,\ lst1))$
$\rightarrow$ qsort-s4p $(s,$ stepn $(s3,\ k),\ a,\ l,\ r,\ n,$ qsort $(1 + last,\ r,\ lst1))$ **endlet**

EVENT: Disable qsort-s3-crock1.

EVENT: Disable qsort-s3-crock2.

THEOREM: qsort-s3-s4-mem
let *lst1* **be** qsort $(l,$ qlast $(l,\ r,\ lst) - 1,$ qpart $(l,\ r,\ lst)),$
last **be** qlast $(l,\ r,\ lst)$
in
(qsort-s3p $(s,\ s3,\ a,\ l,\ r,\ n,\ lst1,\ last)$
$\land$ qsort-statep $(s,\ a,\ l,\ r,\ n,\ lst)$
$\land$ qsort-sk $(s,\ s3,\ a,\ l,\ r,\ n,\ lst)$
$\land$ qsort-sk $(s3,$ stepn $(s3,\ k),\ a,\ 1 + last,\ r,\ n,\ lst1)$
$\land$ disjoint $(a,\ 4 * n,\ x,\ k1)$
$\land$ disjoint (sub (32, qstack $(l,\ r,\ lst)$ − 16, read-sp (s)),
qstack $(l,\ r,\ lst)$,
x,
$k1$))
$\rightarrow$ (read-mem $(x,$ mc-mem (stepn $(s3,\ k)),\ k1)$
$=$ read-mem $(x,$ mc-mem $(s),\ k1))$ **endlet**

EVENT: Disable qstack-la2.

THEOREM: qsort-sk-s-s4
let *lst1* **be** qsort $(l,$ qlast $(l,\ r,\ lst) - 1,$ qpart $(l,\ r,\ lst)),$
last **be** qlast $(l,\ r,\ lst)$
in
(qsort-s3p $(s,\ s3,\ a,\ l,\ r,\ n,\ lst1,\ last)$
$\land$ qsort-statep $(s,\ a,\ l,\ r,\ n,\ lst)$
$\land$ qsort-sk $(s,\ s3,\ a,\ l,\ r,\ n,\ lst)$
$\land$ qsort-sk $(s3,$ stepn $(s3,\ k),\ a,\ 1 + last,\ r,\ n,\ lst1))$
$\rightarrow$ qsort-sk $(s,$ stepn $(s3,\ k),\ a,\ l,\ r,\ n,\ lst)$ **endlet**

EVENT: Disable qsort-s3-s4-mem.

```
; the correctness of the QSORT program.
```

THEOREM: qsort-t–1
qsort-t $(a,\ l,\ \text{idifference}\,(r,\ \texttt{1}),\ n,\ lst) = \text{qsort-t}\,(a,\ l,\ r - 1,\ n,\ lst)$

THEOREM: qsort–1
qsort $(l,\ \text{idifference}\,(r,\ \texttt{1}),\ lst) = \text{qsort}\,(l,\ r - 1,\ lst)$

EVENT: Disable qsort-10.

EVENT: Disable qsort-5.

EVENT: Disable qsort-3.

THEOREM: qsort-correctness-la
$(\text{qsort-statep}\,(s,\ a,\ l,\ r,\ n,\ lst) \land (oplen \leq \texttt{32}) \land \text{d2-7a2-5p}\,(rn))$
$\rightarrow\quad (\text{qsort-s5p}\,(s,\ \text{stepn}\,(s,\ \text{qsort-t}\,(a,\ l,\ r,\ n,\ lst)),\ a,\ l,\ r,\ n,\ \text{qsort}\,(l,\ r,\ lst))$
$\quad\land\quad \text{qsort-sk}\,(s,\ \text{stepn}\,(s,\ \text{qsort-t}\,(a,\ l,\ r,\ n,\ lst)),\ a,\ l,\ r,\ n,\ lst)$
$\quad\land\quad (\text{read-rn}\,(oplen,\ rn,\ \text{mc-rfile}\,(\text{stepn}\,(s,\ \text{qsort-t}\,(a,\ l,\ r,\ n,\ lst))))$
$\quad\quad =\quad \text{read-rn}\,(oplen,\ rn,\ \text{mc-rfile}\,(s))))$

THEOREM: qsort-correctness
let sn **be** $\text{stepn}\,(s,\ \text{qsort-t}\,(a,\ l,\ r,\ n,\ lst))$,
sp **be** $\text{sub}\,(\texttt{32},\ \text{qstack}\,(l,\ r,\ lst) - \texttt{16},\ \text{read-sp}\,(s))$
in
$\text{qsort-statep}\,(s,\ a,\ l,\ r,\ n,\ lst)$
$\rightarrow\quad ((\text{mc-status}\,(sn) = \texttt{'running})$
$\quad\land\quad (\text{mc-pc}\,(sn) = \text{rts-addr}\,(s))$
$\quad\land\quad (\text{read-rn}\,(\texttt{32},\ \texttt{14},\ \text{mc-rfile}\,(sn))$
$\quad\quad =\quad \text{read-rn}\,(\texttt{32},\ \texttt{14},\ \text{mc-rfile}\,(s)))$
$\quad\land\quad (\text{read-rn}\,(\texttt{32},\ \texttt{15},\ \text{mc-rfile}\,(sn))$
$\quad\quad =\quad \text{add}\,(\texttt{32},\ \text{read-rn}\,(\texttt{32},\ \texttt{15},\ \text{mc-rfile}\,(s)),\ \texttt{4}))$
$\quad\land\quad (((oplen \leq \texttt{32}) \land \text{d2-7a2-5p}\,(rn))$
$\quad\quad \rightarrow\quad (\text{read-rn}\,(oplen,\ rn,\ \text{mc-rfile}\,(sn))$
$\quad\quad\quad =\quad \text{read-rn}\,(oplen,\ rn,\ \text{mc-rfile}\,(s))))$
$\quad\land\quad ((\text{disjoint}\,(sp,\ \text{qstack}\,(l,\ r,\ lst),\ x,\ k)$
$\quad\quad \land\quad \text{disjoint}\,(a,\ \texttt{4} * n,\ x,\ k))$
$\quad\quad \rightarrow\quad (\text{read-mem}\,(x,\ \text{mc-mem}\,(sn),\ k)$
$\quad\quad\quad =\quad \text{read-mem}\,(x,\ \text{mc-mem}\,(s),\ k)))$
$\quad\land\quad \text{mem-ilst}\,(\texttt{4},\ a,\ \text{mc-mem}\,(sn),\ n,\ \text{qsort}\,(l,\ r,\ lst)))$ **endlet**

```
; in the logic, qsort is correct:
;       1. if left <= i <= j <= right,  lst'[i] <= lst'[j].
;       2. for all x,  (count-lst x l r lst) = (count-lst x l r lst').
```

THEOREM: put-commute
$(\text{fix}(i) \neq \text{fix}(j))$
$\rightarrow \quad (\text{put-nth}(\mathit{v1}, i, \text{put-nth}(\mathit{v2}, j, \mathit{lst})) = \text{put-nth}(\mathit{v2}, j, \text{put-nth}(\mathit{v1}, i, \mathit{lst})))$

THEOREM: swap-put-commute
$((i < l) \wedge (i < r))$
$\rightarrow \quad (\text{swap}(l, r, \text{put-nth}(v, i, \mathit{lst})) = \text{put-nth}(v, i, \text{swap}(l, r, \mathit{lst})))$

THEOREM: swap-commute
$((i < l) \wedge (i < r) \wedge (j < l) \wedge (j < r))$
$\rightarrow \quad (\text{swap}(l, r, \text{swap}(i, j, \mathit{lst})) = \text{swap}(i, j, \text{swap}(l, r, \mathit{lst})))$

DEFINITION:
$\text{sublst-ileq}(x, l, r, \mathit{lst})$
$= \quad \textbf{if } r < l \textbf{ then t}$
$\quad\quad \textbf{else } \text{ileq}(x, \text{get-nth}(l, \mathit{lst})) \wedge \text{sublst-ileq}(x, 1 + l, r, \mathit{lst}) \textbf{ endif}$

DEFINITION:
$\text{sublsts-ileq}(l, r, \mathit{lst}, \mathit{l1}, \mathit{r1}, \mathit{lst1})$
$= \quad \textbf{if } r < l \textbf{ then t}$
$\quad\quad \textbf{else } \text{sublst-ileq}(\text{get-nth}(l, \mathit{lst}), \mathit{l1}, \mathit{r1}, \mathit{lst1})$
$\quad\quad\quad\quad \wedge \quad \text{sublsts-ileq}(1 + l, r, \mathit{lst}, \mathit{l1}, \mathit{r1}, \mathit{lst1}) \textbf{ endif}$

THEOREM: sublst-ileq-lemma
$(\text{sublst-ileq}(x, l, r, \mathit{lst}) \wedge (l \leq j) \wedge (j \leq r))$
$\rightarrow \quad (\neg\, \text{ilessp}(\text{get-nth}(j, \mathit{lst}), x))$

THEOREM: sublsts-ileq-la1
$(\text{sublsts-ileq}(l, r, \mathit{lst}, \mathit{l1}, \mathit{r1}, \mathit{lst1}) \wedge (l \leq i) \wedge (i \leq r))$
$\rightarrow \quad \text{sublst-ileq}(\text{get-nth}(i, \mathit{lst}), \mathit{l1}, \mathit{r1}, \mathit{lst1})$

THEOREM: sublst-ileq-put
$(\text{ileq}(x, y) \wedge \text{ileq}(x, \text{get-nth}(j, \mathit{lst})))$
$\rightarrow \quad (\text{sublst-ileq}(x, l, r, \text{put-nth}(y, j, \mathit{lst})) = \text{sublst-ileq}(x, l, r, \mathit{lst}))$

THEOREM: sublsts-ileq-put
$(\text{sublst-ileq}(y, \mathit{l1}, \mathit{r1}, \mathit{lst1}) \wedge \text{sublst-ileq}(\text{get-nth}(j, \mathit{lst}), \mathit{l1}, \mathit{r1}, \mathit{lst1}))$
$\rightarrow \quad (\text{sublsts-ileq}(l, r, \text{put-nth}(y, j, \mathit{lst}), \mathit{l1}, \mathit{r1}, \mathit{lst1})$
$\quad\quad = \quad \text{sublsts-ileq}(l, r, \mathit{lst}, \mathit{l1}, \mathit{r1}, \mathit{lst1}))$

THEOREM: sublst-ileq-swap-la
$(\text{sublst-ileq}(x, l, r, \mathit{lst})$
$\wedge \quad (l \leq i)$
$\wedge \quad (i \leq r)$
$\wedge \quad (l \leq j)$
$\wedge \quad (j \leq r))$
$\rightarrow \quad \text{sublst-ileq}(x, l, r, \text{swap}(i, j, \mathit{lst}))$

THEOREM: sublsts-ileq-swap-la
(sublsts-ileq $(l,\ r,\ lst,\ l1,\ r1,\ lst1)$
$\land\quad (l \leq i)$
$\land\quad (i \leq r)$
$\land\quad (l \leq j)$
$\land\quad (j \leq r))$
$\rightarrow$ sublsts-ileq $(l,\ r,$ swap $(i,\ j,\ lst),\ l1,\ r1,\ lst1)$

EVENT: Disable swap.

THEOREM: sublst-ileq-swap-swap
sublst-ileq $(x,\ l,\ r,$ swap $(i,\ j,$ swap $(i,\ j,\ lst))) =$ sublst-ileq $(x,\ l,\ r,\ lst)$

THEOREM: sublst-ileq-swap
$((l \leq i) \land (i \leq r) \land (l \leq j) \land (j \leq r))$
$\rightarrow$ (sublst-ileq $(x,\ l,\ r,$ swap $(i,\ j,\ lst)) =$ sublst-ileq $(x,\ l,\ r,\ lst))$

THEOREM: sublsts-ileq-swap-swap
sublsts-ileq $(l,\ r,$ swap $(i,\ j,$ swap $(i,\ j,\ lst)),\ l1,\ r1,\ lst1)$
$=$ sublsts-ileq $(l,\ r,\ lst,\ l1,\ r1,\ lst1)$

THEOREM: sublsts-ileq-swap
$((l \leq i) \land (i \leq r) \land (l \leq j) \land (j \leq r))$
$\rightarrow$ (sublsts-ileq $(l,\ r,$ swap $(i,\ j,\ lst),\ l1,\ r1,\ lst1)$
$=$ sublsts-ileq $(l,\ r,\ lst,\ l1,\ r1,\ lst1))$

THEOREM: sublst-ileq-qpart-aux
$((l \leq last)$
$\land\quad (last < i)$
$\land\quad (last \leq r)$
$\land\quad (l \leq l1)$
$\land\quad (l1 \leq r)$
$\land\quad (r1 \leq r))$
$\rightarrow$ (sublst-ileq $(x,\ l,\ r,$ qpart-aux $(l1,\ r1,\ lst,\ last,\ i))$
$=$ sublst-ileq $(x,\ l,\ r,\ lst))$

THEOREM: sublsts-ileq-qpart-aux
$((l \leq last)$
$\land\quad (last < i)$
$\land\quad (last \leq r2)$
$\land\quad (l \leq l2)$
$\land\quad (l2 \leq r)$
$\land\quad (r2 \leq r))$
$\rightarrow$ (sublsts-ileq $(l,\ r,$ qpart-aux $(l2,\ r2,\ lst,\ last,\ i),\ l1,\ r1,\ lst1)$
$=$ sublsts-ileq $(l,\ r,\ lst,\ l1,\ r1,\ lst1))$

THEOREM: sublst-ileq-qsort
$((l \leq l1) \land (r1 \leq r))$
$\rightarrow \quad (\text{sublst-ileq}\,(x,\, l,\, r,\, \text{qsort}\,(l1,\, r1,\, lst)) = \text{sublst-ileq}\,(x,\, l,\, r,\, lst))$

THEOREM: sublsts-ileq-qsort1
$((l \leq l2) \land (r2 \leq r))$
$\rightarrow \quad (\text{sublsts-ileq}\,(l,\, r,\, \text{qsort}\,(l2,\, r2,\, lst),\, l1,\, r1,\, lst1)$
$\quad = \quad \text{sublsts-ileq}\,(l,\, r,\, lst,\, l1,\, r1,\, lst1))$

THEOREM: sublsts-ileq-qsort2
$((l1 \leq l2) \land (r2 \leq r1))$
$\rightarrow \quad (\text{sublsts-ileq}\,(l,\, r,\, lst,\, l1,\, r1,\, \text{qsort}\,(l2,\, r2,\, lst1))$
$\quad = \quad \text{sublsts-ileq}\,(l,\, r,\, lst,\, l1,\, r1,\, lst1))$

DEFINITION:
qpartx $(l,\, r,\, lst,\, last,\, i)$
$=$ **if** $r < i$ **then** lst
elseif ilessp (get-nth $(i,\, lst)$, get-nth $(l,\, lst)$)
then qpartx $(l,\, r,$ swap $(1 + last,\, i,\, lst),\, 1 + last,\, 1 + i)$
else qpartx $(l,\, r,\, lst,\, last,\, 1 + i)$ **endif**

THEOREM: qpart-aux-qpartx
qpart-aux $(l,\, r,\, lst,\, last,\, i)$
$=$ swap $(l,$ qlast-aux $(l,\, r,\, lst,\, last,\, i)$, qpartx $(l,\, r,\, lst,\, last,\, i))$

THEOREM: qpartx-get-1
$((last < i) \land (right < j))$
$\rightarrow \quad (\text{get-nth}\,(j,\, \text{qpartx}\,(left,\, right,\, lst,\, last,\, i)) = \text{get-nth}\,(j,\, lst))$

THEOREM: qpartx-get-2
$((last < i) \land (j \leq last))$
$\rightarrow \quad (\text{get-nth}\,(j,\, \text{qpartx}\,(left,\, right,\, lst,\, last,\, i)) = \text{get-nth}\,(j,\, lst))$

THEOREM: qpartx-ilessp-1
$((l \leq last)$
$\land \quad (last < i)$
$\land \quad (last < j)$
$\land \quad (j \leq \text{qlast-aux}\,(l,\, r,\, lst,\, last,\, i)))$
$\rightarrow \quad \text{ilessp}\,(\text{get-nth}\,(j,\, \text{qpartx}\,(l,\, r,\, lst,\, last,\, i)),\, \text{get-nth}\,(l,\, lst))$

DEFINITION:
open-sublst-ileq $(x,\, l,\, r,\, lst)$
$=$ **if** $(1 + l) < r$
then ileq $(x,$ get-nth $(1 + l,\, lst)) \land$ open-sublst-ileq $(x,\, 1 + l,\, r,\, lst)$
else t endif

THEOREM: open-sublst-ileq-la0
$((1 + l) \not< r) \rightarrow \text{open-sublst-ileq}\,(x,\ l,\ r,\ \mathit{lst})$

THEOREM: open-sublst-ileq-la1
$(\text{open-sublst-ileq}\,(x,\ \mathit{last},\ i,\ \mathit{lst}) \land (\mathit{last} < j) \land (j < i))$
$\rightarrow \quad (\neg\ \text{ilessp}\,(\text{get-nth}\,(j,\ \mathit{lst}),\ x))$

THEOREM: open-sublst-ileq-la2
$\text{open-sublst-ileq}\,(x,\ \mathit{last},\ 1 + i,\ \mathit{lst})$
$= \quad \textbf{if}\ \text{ileq}\,(x,\ \text{get-nth}\,(i,\ \mathit{lst}))\ \ \textbf{then}\ \text{open-sublst-ileq}\,(x,\ \mathit{last},\ i,\ \mathit{lst})$
$\qquad \textbf{else}\ (1 + \mathit{last}) \not< (1 + i)\ \textbf{endif}$

THEOREM: open-sublst-ileq-la3
$(j \leq l)$
$\rightarrow \quad (\text{open-sublst-ileq}\,(x,\ l,\ r,\ \text{put-nth}\,(v,\ j,\ \mathit{lst}))$
$\qquad = \quad \text{open-sublst-ileq}\,(x,\ l,\ r,\ \mathit{lst}))$

THEOREM: open-sublst-ileq-la4
$(r \leq j)$
$\rightarrow \quad (\text{open-sublst-ileq}\,(x,\ l,\ r,\ \text{put-nth}\,(v,\ j,\ \mathit{lst}))$
$\qquad = \quad \text{open-sublst-ileq}\,(x,\ l,\ r,\ \mathit{lst}))$

THEOREM: open-sublst-ileq-la5
$\text{open-sublst-ileq}\,(x,\ 1 + \mathit{last},\ 1 + i,\ \text{swap}\,(1 + \mathit{last},\ i,\ \mathit{lst}))$
$= \quad \text{open-sublst-ileq}\,(x,\ \mathit{last},\ i,\ \mathit{lst})$

THEOREM: qpartx-ilessp-2
$(\text{open-sublst-ileq}\,(\text{get-nth}\,(l,\ \mathit{lst}),\ \mathit{last},\ i,\ \mathit{lst})$
$\land \quad (l \leq \mathit{last})$
$\land \quad (\mathit{last} < i)$
$\land \quad (\text{qlast-aux}\,(l,\ r,\ \mathit{lst},\ \mathit{last},\ i) < j)$
$\land \quad (j \leq r))$
$\rightarrow \quad (\neg\ \text{ilessp}\,(\text{get-nth}\,(j,\ \text{qpartx}\,(l,\ r,\ \mathit{lst},\ \mathit{last},\ i)),\ \text{get-nth}\,(l,\ \mathit{lst})))$

THEOREM: open-sublst-ileq-la6
$\text{open-sublst-ileq}\,(\text{get-nth}\,(l,\ \mathit{lst}),\ l,\ 1 + l,\ \mathit{lst})$

THEOREM: qpart-ilessp-la1-la
$((l \leq j) \land (j < \text{qlast-aux}\,(l,\ r,\ \mathit{lst},\ l,\ 1 + l)))$
$\rightarrow \quad (\neg\ \text{ilessp}\,(\text{get-nth}\,(l,\ \mathit{lst}),\ \text{get-nth}\,(j,\ \text{qpart-aux}\,(l,\ r,\ \mathit{lst},\ l,\ 1 + l))))$

THEOREM: qpart-ilessp-la1
$((l \leq j) \land (j \leq (\text{qlast-aux}\,(l,\ r,\ \mathit{lst},\ l,\ 1 + l) - 1)))$
$\rightarrow \quad (\neg\ \text{ilessp}\,(\text{get-nth}\,(l,\ \mathit{lst}),\ \text{get-nth}\,(j,\ \text{qpart-aux}\,(l,\ r,\ \mathit{lst},\ l,\ 1 + l))))$

THEOREM: qpart-ilessp-la2
$((\text{qlast-aux}\,(l,\ r,\ \mathit{lst},\ l,\ 1 + l) \le j) \land (j \le r))$
$\rightarrow\quad (\neg\ \text{ilessp}\,(\text{get-nth}\,(j,\ \text{qpart-aux}\,(l,\ r,\ \mathit{lst},\ l,\ 1 + l)),\ \text{get-nth}\,(l,\ \mathit{lst})))$

THEOREM: qpart-equal
$((l \le \mathit{last}) \land (\mathit{last} < i))$
$\rightarrow\quad (\text{get-nth}\,(\text{qlast-aux}\,(l,\ r,\ \mathit{lst},\ \mathit{last},\ i),\ \text{qpart-aux}\,(l,\ r,\ \mathit{lst},\ \mathit{last},\ i))$
$\qquad = \quad \text{get-nth}\,(l,\ \mathit{lst}))$

THEOREM: qsort-get-1
$(j < \mathit{left}) \rightarrow (\text{get-nth}\,(j,\ \text{qsort}\,(\mathit{left},\ \mathit{right},\ \mathit{lst})) = \text{get-nth}\,(j,\ \mathit{lst}))$

THEOREM: get-swap-1
$((j < i) \land (k < i)) \rightarrow (\text{get-nth}\,(i,\ \text{swap}\,(j,\ k,\ \mathit{lst})) = \text{get-nth}\,(i,\ \mathit{lst}))$

THEOREM: qlast-aux-swap
$((l \le \mathit{last}) \land (\mathit{last} < i) \land (a < l) \land (b < l))$
$\rightarrow\quad (\text{qlast-aux}\,(l,\ r,\ \text{swap}\,(a,\ b,\ \mathit{lst}),\ \mathit{last},\ i) = \text{qlast-aux}\,(l,\ r,\ \mathit{lst},\ \mathit{last},\ i))$

THEOREM: qlast-swap
$((l < r) \land (a < l) \land (b < l))$
$\rightarrow\quad (\text{qlast}\,(l,\ r,\ \text{swap}\,(a,\ b,\ \mathit{lst})) = \text{qlast}\,(l,\ r,\ \mathit{lst}))$

THEOREM: qpart-aux-swap
$((l \le \mathit{last}) \land (\mathit{last} < i) \land (a < l) \land (b < l))$
$\rightarrow\quad (\text{qpart-aux}\,(l,\ r,\ \text{swap}\,(a,\ b,\ \mathit{lst}),\ \mathit{last},\ i)$
$\qquad = \quad \text{swap}\,(a,\ b,\ \text{qpart-aux}\,(l,\ r,\ \mathit{lst},\ \mathit{last},\ i)))$

THEOREM: qpart-swap
$((l < r) \land (a < l) \land (b < l))$
$\rightarrow\quad (\text{qpart}\,(l,\ r,\ \text{swap}\,(a,\ b,\ \mathit{lst})) = \text{swap}\,(a,\ b,\ \text{qpart}\,(l,\ r,\ \mathit{lst})))$

THEOREM: qsort-swap
$((i < l) \land (j < l))$
$\rightarrow\quad (\text{qsort}\,(l,\ r,\ \text{swap}\,(i,\ j,\ \mathit{lst})) = \text{swap}\,(i,\ j,\ \text{qsort}\,(l,\ r,\ \mathit{lst})))$

THEOREM: qsort-qpartx
$((l \le r) \land (l \le \mathit{last}) \land (\mathit{last} < i) \land (\mathit{last} \le r) \land (r < \mathit{l1}))$
$\rightarrow\quad (\text{qsort}\,(\mathit{l1},\ \mathit{r1},\ \text{qpartx}\,(l,\ r,\ \mathit{lst},\ \mathit{last},\ i))$
$\qquad = \quad \text{qpartx}\,(l,\ r,\ \text{qsort}\,(\mathit{l1},\ \mathit{r1},\ \mathit{lst}),\ \mathit{last},\ i))$

THEOREM: qsort-get-2
$(r < \mathit{l1})$
$\rightarrow\quad (\text{get-nth}\,(j,\ \text{qsort}\,(\mathit{l1},\ \mathit{r1},\ \text{qsort}\,(l,\ r,\ \mathit{lst})))$
$\qquad = \quad \textbf{if}\ r < j\ \ \textbf{then}\ \text{get-nth}\,(j,\ \text{qsort}\,(\mathit{l1},\ \mathit{r1},\ \mathit{lst}))$
$\qquad\qquad \textbf{else}\ \text{get-nth}\,(j,\ \text{qsort}\,(l,\ r,\ \mathit{lst}))\ \textbf{endif})$

EVENT: Disable qpart-aux-qpartx.

THEOREM: ilessp-trans
$(\text{ilessp}\,(a,\ b) \land \text{ileq}\,(b,\ c)) \rightarrow \text{ilessp}\,(a,\ c)$

THEOREM: qpart-ilessp
$$
\begin{array}{ll}
((l \le i) & \\
\land & (i \le \text{qlast}\,(l,\ r,\ \mathit{lst})) \\
\land & (\text{qlast}\,(l,\ r,\ \mathit{lst}) \le j) \\
\land & (j \le r)) \\
\rightarrow & (\neg\ \text{ilessp}\,(\text{get-nth}\,(j,\ \text{qpart}\,(l,\ r,\ \mathit{lst})),\ \text{get-nth}\,(i,\ \text{qpart}\,(l,\ r,\ \mathit{lst}))))
\end{array}
$$

THEOREM: qsort-get3
$$
\begin{array}{ll}
\multicolumn{2}{l}{\text{get-nth}\,(j,\ \text{qsort}\,(1 + \mathit{last},\ r,\ \text{qsort}\,(l,\ \mathit{last} - 1,\ \mathit{lst})))} \\
= & \textbf{if}\ j \le (\mathit{last} - 1)\ \ \textbf{then}\ \text{get-nth}\,(j,\ \text{qsort}\,(l,\ \mathit{last} - 1,\ \mathit{lst})) \\
 & \textbf{else}\ \text{get-nth}\,(j,\ \text{qsort}\,(1 + \mathit{last},\ r,\ \mathit{lst}))\ \textbf{endif}
\end{array}
$$

THEOREM: sublsts-ileq-la2
$$
\begin{array}{ll}
\multicolumn{2}{l}{(\text{sublsts-ileq}\,(l,\ r,\ \mathit{lst},\ \mathit{l1},\ \mathit{r1},\ \mathit{lst1})} \\
\land & (l \le i) \\
\land & (i \le r) \\
\land & (\mathit{l1} \le j) \\
\land & (j \le \mathit{r1})) \\
\rightarrow & (\neg\ \text{ilessp}\,(\text{get-nth}\,(j,\ \mathit{lst1}),\ \text{get-nth}\,(i,\ \mathit{lst})))
\end{array}
$$

THEOREM: qpart-ilessp-closed-1
$$
\begin{array}{ll}
\multicolumn{2}{l}{(\text{qlast}\,(l,\ r,\ \mathit{lst}) \le \mathit{last})} \\
\rightarrow & \text{sublst-ileq}\,(\text{get-nth}\,(\text{qlast}\,(l,\ r,\ \mathit{lst}),\ \text{qpart}\,(l,\ r,\ \mathit{lst})), \\
 & \qquad\qquad \mathit{last}, \\
 & \qquad\qquad r, \\
 & \qquad\qquad \text{qpart}\,(l,\ r,\ \mathit{lst}))
\end{array}
$$

THEOREM: qsort-ilessp-1
$$
\begin{array}{l}
\textbf{let}\ \mathit{lst1}\ \ \textbf{be}\ \ \text{qpart}\,(l,\ r,\ \mathit{lst}), \\
\qquad \mathit{last}\ \ \textbf{be}\ \ \text{qlast}\,(l,\ r,\ \mathit{lst}) \\
\textbf{in} \\
((\mathit{last} \le j) \land (j \le r)) \\
\rightarrow \quad (\neg\ \text{ilessp}\,(\text{get-nth}\,(j,\ \text{qsort}\,(1 + \mathit{last},\ r,\ \mathit{lst1})), \\
\qquad\qquad\qquad \text{get-nth}\,(\mathit{last},\ \mathit{lst1})))\ \textbf{endlet}
\end{array}
$$

THEOREM: qpart-ilessp-closed-2
$$
\begin{array}{ll}
\multicolumn{2}{l}{\textbf{let}\ \mathit{lst1}\ \ \textbf{be}\ \ \text{qpart}\,(l,\ r,\ \mathit{lst})} \\
\multicolumn{2}{l}{\textbf{in}} \\
\multicolumn{2}{l}{((\text{qlast}\,(l,\ r,\ \mathit{lst}) \le \mathit{last})} \\
\land & (l \le j)
\end{array}
$$

$\wedge \quad (j \leq (\text{qlast}(l,\ r,\ lst) - 1)))$
$\rightarrow \quad \text{sublst-ileq}(\text{get-nth}(j,\ lst1),\ last,\ r,\ lst1)$ **endlet**

THEOREM: qpart-ilessp-closed-3
let $last$ **be** $\text{qlast}(l,\ r,\ lst)$,
$lst1$ **be** $\text{qpart}(l,\ r,\ lst)$
in
$(l \leq l1) \rightarrow \text{sublsts-ileq}(l1,\ last - 1,\ lst1,\ last,\ r,\ lst1)$ **endlet**

THEOREM: qsort-ilessp-2
let $lst1$ **be** $\text{qpart}(l,\ r,\ lst)$,
$last$ **be** $\text{qlast}(l,\ r,\ lst)$
in
$((l \leq i) \wedge (i \leq (last - 1)) \wedge (last \leq j) \wedge (j \leq r))$
$\rightarrow \quad (\neg\ \text{ilessp}(\text{get-nth}(j,\ \text{qsort}(1 + last,\ r,\ lst1)),$
$\text{get-nth}(i,\ \text{qsort}(l,\ last - 1,\ lst1))))$ **endlet**

THEOREM: qsort-ordered
$((left \leq i) \wedge (i \leq j) \wedge (j \leq right))$
$\rightarrow \quad (\neg\ \text{ilessp}(\text{get-nth}(j,\ \text{qsort}(left,\ right,\ lst)),$
$\text{get-nth}(i,\ \text{qsort}(left,\ right,\ lst))))$

DEFINITION:
$\text{orderedp1}(l,\ r,\ lst)$
$=$ **if** $r \leq l$ **then** **t**
else $\text{ileq}(\text{get-nth}(l,\ lst),\ \text{get-nth}(1 + l,\ lst))$
$\wedge \quad \text{orderedp1}(1 + l,\ r,\ lst)$ **endif**

THEOREM: qsort-orderedp1-la
$(left \leq left1) \rightarrow \text{orderedp1}(left1,\ right,\ \text{qsort}(left,\ right,\ lst))$

THEOREM: qsort-orderedp1
$\text{orderedp1}(left,\ right,\ \text{qsort}(left,\ right,\ lst))$

DEFINITION: $\text{transwap}(i,\ lst) = \text{swap}(i,\ 1 + i,\ lst)$

DEFINITION:
$\text{lst-eq}(l,\ r,\ lst,\ lst1)$
$=$ **if** $r < l$ **then** **t**
else $(\text{get-nth}(l,\ lst) = \text{get-nth}(l,\ lst1))$
$\wedge \quad \text{lst-eq}(1 + l,\ r,\ lst,\ lst1)$ **endif**

DEFINITION:
$\text{count-lst}(x,\ l,\ r,\ lst)$
$=$ **if** $r < l$ **then** `0`
elseif $x = \text{get-nth}(l,\ lst)$ **then** $1 + \text{count-lst}(x,\ 1 + l,\ r,\ lst)$
else $\text{count-lst}(x,\ 1 + l,\ r,\ lst)$ **endif**

THEOREM: count-lst-0
$(l \notin \mathbf{N}) \rightarrow (\text{count-lst}\,(x,\ l,\ r,\ lst) = \text{count-lst}\,(x,\ \mathtt{0},\ r,\ lst))$

THEOREM: count-swapii
$\text{count-lst}\,(x,\ l,\ r,\ \text{swap}\,(i,\ i,\ lst)) = \text{count-lst}\,(x,\ l,\ r,\ lst)$

THEOREM: count-lst-put-1
$(i < l) \rightarrow (\text{count-lst}\,(x,\ l,\ r,\ \text{put-nth}\,(v,\ i,\ lst)) = \text{count-lst}\,(x,\ l,\ r,\ lst))$

THEOREM: count-lst-swap-1
$((i < l) \wedge (j < l))$
$\rightarrow \quad (\text{count-lst}\,(x,\ l,\ r,\ \text{swap}\,(i,\ j,\ lst)) = \text{count-lst}\,(x,\ l,\ r,\ lst))$

THEOREM: count-transwap-0
$(l < r) \rightarrow (\text{count-lst}\,(x,\ l,\ r,\ \text{swap}\,(l,\ 1 + l,\ lst)) = \text{count-lst}\,(x,\ l,\ r,\ lst))$

THEOREM: swap-0
$(i \notin \mathbf{N})$
$\rightarrow \quad ((\text{swap}\,(i,\ j,\ lst) = \text{swap}\,(\mathtt{0},\ j,\ lst))$
$\quad\quad \wedge \quad (\text{swap}\,(j,\ i,\ lst) = \text{swap}\,(j,\ \mathtt{0},\ lst)))$

THEOREM: count-transwap
$((l \leq i) \wedge (i < r))$
$\rightarrow \quad (\text{count-lst}\,(x,\ l,\ r,\ \text{transwap}\,(i,\ lst)) = \text{count-lst}\,(x,\ l,\ r,\ lst))$

THEOREM: swap-rec-la
$(i \leq j)$
$\rightarrow \quad \text{lst-eq}\,(l,\ r,\ \text{swap}\,(i,\ 1 + j,\ lst),\ \text{swap}\,(i,\ j,\ \text{swap}\,(j,\ 1 + j,\ \text{swap}\,(i,\ j,\ lst))))$

THEOREM: count-lst-eq
$\text{lst-eq}\,(l,\ r,\ lst,\ lst1) \rightarrow (\text{count-lst}\,(x,\ l,\ r,\ lst) = \text{count-lst}\,(x,\ l,\ r,\ lst1))$

THEOREM: count-lst-swap-rec
$(i \leq j)$
$\rightarrow \quad (\text{count-lst}\,(x,\ l,\ r,\ \text{swap}\,(i,\ 1 + j,\ lst))$
$\quad\quad = \quad \text{count-lst}\,(x,\ l,\ r,\ \text{swap}\,(i,\ j,\ \text{transwap}\,(j,\ \text{swap}\,(i,\ j,\ lst)))))$

DEFINITION:
$\text{swap-induct}\,(i,\ j,\ lst)$
$= \quad$ **if** $j \leq i$ **then t**
else $\text{swap-induct}\,(i,\ j - 1,\ \text{transwap}\,(j - 1,\ \text{swap}\,(i,\ j - 1,\ lst)))$
$\quad \wedge \quad \text{swap-induct}\,(i,\ j - 1,\ lst)$ **endif**

THEOREM: count-lst-swap
$((l \leq i) \wedge (i \leq j) \wedge (j \leq r))$
$\rightarrow \quad (\text{count-lst}\,(x,\ l,\ r,\ \text{swap}\,(i,\ j,\ lst)) = \text{count-lst}\,(x,\ l,\ r,\ lst))$

Theorem: count-lst-qpart-aux

$$
\begin{array}{ll}
((\mathit{l1} \le \mathit{last}) & \\
\ \wedge \quad (\mathit{last} < i) & \\
\ \wedge \quad (\mathit{last} \le \mathit{r1}) & \\
\ \wedge \quad (l \le \mathit{l1}) & \\
\ \wedge \quad (\mathit{l1} \le r) & \\
\ \wedge \quad (\mathit{r1} \le r)) & \\
\rightarrow \quad (\text{count-lst}\,(x,\ l,\ r,\ \text{qpart-aux}\,(\mathit{l1},\ \mathit{r1},\ \mathit{lst},\ \mathit{last},\ i)) & \\
\qquad = \quad \text{count-lst}\,(x,\ l,\ r,\ \mathit{lst})) &
\end{array}
$$

Theorem: count-lst-qsort-la

$$
\begin{array}{l}
((l \le \mathit{l1}) \wedge (\mathit{r1} \le r)) \\
\rightarrow \quad (\text{count-lst}\,(x,\ l,\ r,\ \text{qsort}\,(\mathit{l1},\ \mathit{r1},\ \mathit{lst})) = \text{count-lst}\,(x,\ l,\ r,\ \mathit{lst}))
\end{array}
$$

Theorem: count-lst-qsort

$$\text{count-lst}\,(x,\ l,\ r,\ \text{qsort}\,(l,\ r,\ \mathit{lst})) = \text{count-lst}\,(x,\ l,\ r,\ \mathit{lst})$$

Index