```
#|

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

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

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

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

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

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

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

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

EVENT: Disable qlast.

EVENT: Disable qpart.

DEFINITION:
qsort $(l,\ r,\ lst)$
$=$ **if** $l < r$
then qsort $(1 + \text{qlast}\,(l,\ r,\ lst),$
$r,$
$\text{qsort}\,(l,\ \text{qlast}\,(l,\ r,\ lst) - 1,\ \text{qpart}\,(l,\ r,\ lst)))$
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) ÷ 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)
     ∧  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) ≮ 68
```

THEOREM: qstack-la1

```
let last  be  qlast (l, r, lst),
    lst1  be  qpart (l, r, lst)
in
(l < r)
→   (qstack (l, r, lst) ≮ (40 + qstack (l, last − 1, lst1))) endlet
```

THEOREM: qstack-la2

```
let last  be  qlast (l, r, lst),
```

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

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

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

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

THEOREM: qstack-ubound
qstack $(l, r, lst) \leq (68 + (52 * (r - l)))$

EVENT: Disable qstack.

```
; the initial state.
```

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

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

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

$\wedge$ ($last$ = nat-to-int (read-rn (32, 2, mc-rfile ($s0$)), 32))
$\wedge$ (i = nat-to-int (read-rn (32, 0, mc-rfile ($s0$)), 32))
$\wedge$ ($l \in \mathbf{N}$)
$\wedge$ ($r \in \mathbf{N}$)
$\wedge$ ($n \in \mathbf{N}$)
$\wedge$ ($last \in \mathbf{N}$)
$\wedge$ ($i \in \mathbf{N}$)
$\wedge$ ($l < r$)
$\wedge$ ($last < i$)
$\wedge$ ($i \leq (1 + r)$)
$\wedge$ ($r < n$)
$\wedge$ int-rangep (n, 32)
$\wedge$ uint-rangep (4 $*$ n, 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:
qsort-s1p (s, $s1$, a, l, r, n, lst, $last$)
= ((linked-rts-addr ($s1$) = rts-addr (s))
$\wedge$ (linked-a6 ($s1$) = read-an (32, 6, s))
$\wedge$ equal* (read-rn (32, 14, mc-rfile ($s1$)), sub (32, 4, read-sp (s)))
$\wedge$ equal* (read-rn (32, 15, mc-rfile ($s1$)), sub (32, 40, read-sp (s)))
$\wedge$ (movem-saved ($s1$, 4, 20, 5)
= readm-rn (32, '(2 3 4 10 11), mc-rfile (s)))
$\wedge$ (mc-status ($s1$) = 'running)
$\wedge$ (mc-pc ($s1$) = read-an (32, 2, $s1$))
$\wedge$ evenp (read-an (32, 2, $s1$))
$\wedge$ evenp (rts-addr ($s1$))
$\wedge$ rom-addrp (read-an (32, 2, $s1$), mc-mem ($s1$), 138)
$\wedge$ mcode-addrp (read-an (32, 2, $s1$), mc-mem ($s1$), QSORT-CODE)
$\wedge$ rom-addrp (sub (32, 116, rts-addr ($s1$)), mc-mem ($s1$), 138)
$\wedge$ mcode-addrp (sub (32, 116, rts-addr ($s1$)), mc-mem ($s1$), QSORT-CODE)
$\wedge$ ram-addrp (sub (32, 52, read-sp (s)), mc-mem ($s1$), 68)
$\wedge$ ram-addrp (a, mc-mem ($s1$), 4 $*$ n)
$\wedge$ mem-ilst (4, a, mc-mem ($s1$), n, lst)
$\wedge$ disjoint (a, 4 $*$ n, sub (32, 52, read-sp (s)), 68)
$\wedge$ (a = read-an (32, 3, $s1$))
$\wedge$ (l = nat-to-int (read-rn (32, 3, mc-rfile ($s1$)), 32))
$\wedge$ (r = nat-to-int (read-rn (32, 4, mc-rfile ($s1$)), 32))
$\wedge$ ($last$ = nat-to-int (read-rn (32, 2, mc-rfile ($s1$)), 32))
$\wedge$ (a = read-mem (add (32, read-sp ($s1$), 4), mc-mem ($s1$), 4))
$\wedge$ (l = iread-mem (add (32, read-sp ($s1$), 8), mc-mem ($s1$), 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))
  ∧ (linked-a6 ($s3$) = read-an (32, 6, s))
  ∧ equal* (read-rn (32, 14, mc-rfile ($s3$)), sub (32, 4, read-sp (s)))
  ∧ equal* (read-rn (32, 15, mc-rfile ($s3$)), sub (32, 52, read-sp (s)))
  ∧ (movem-saved ($s3$, 4, 20, 5)
      = readm-rn (32, '(2 3 4 10 11), mc-rfile (s)))
  ∧ (mc-status ($s3$) = 'running)
  ∧ (mc-pc ($s3$) = read-an (32, 2, $s3$))
  ∧ evenp (read-an (32, 2, $s3$))
  ∧ evenp (rts-addr ($s3$))
  ∧ rom-addrp (read-an (32, 2, $s3$), mc-mem ($s3$), 138)
  ∧ mcode-addrp (read-an (32, 2, $s3$), mc-mem ($s3$), QSORT-CODE)
  ∧ rom-addrp (sub (32, 128, rts-addr ($s3$)), mc-mem ($s3$), 138)
  ∧ mcode-addrp (sub (32, 128, rts-addr ($s3$)), mc-mem ($s3$), QSORT-CODE)
  ∧ ram-addrp (read-sp ($s3$), mc-mem ($s3$), 68)
  ∧ ram-addrp (a, mc-mem ($s3$), $4 * n$)
  ∧ mem-ilst (4, a, mc-mem ($s3$), n, lst)
  ∧ disjoint (a, $4 * n$, read-sp ($s3$), 68)
  ∧ (a = read-an (32, 3, $s3$))
  ∧ (l = nat-to-int (read-rn (32, 3, mc-rfile ($s3$)), 32))
  ∧ (r = nat-to-int (read-rn (32, 4, mc-rfile ($s3$)), 32))
  ∧ ($last$ = nat-to-int (read-rn (32, 2, mc-rfile ($s3$)), 32))
  ∧ (a = read-mem (add (32, read-sp ($s3$), 4), mc-mem ($s3$), 4))
  ∧ (iread-mem (add (32, read-sp ($s3$), 8), mc-mem ($s3$), 4)
      = (1 + $last$))
  ∧ (r = iread-mem (add (32, read-sp ($s3$), 12), mc-mem ($s3$), 4))
  ∧ ($l \in \mathbf{N}$)
  ∧ ($r \in \mathbf{N}$)
  ∧ ($last \in \mathbf{N}$)
  ∧ ($l < r$)
  ∧ ($last \leq r$)
  ∧ ($r < n$)
  ∧ int-rangep (n, 32)
  ∧ 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)
↔ ∀ 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** $\text{swap}(l, (l + r) \div 2, lst)$
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
$(\text{qsort-sp}(s, a, l, r, n, lst) \wedge (l < r) \wedge \text{d5-7a4-5p}(rn))$
$\rightarrow \quad (\text{read-rn}(oplen, rn, \text{mc-rfile}(\text{stepn}(s, 18)))$
$\quad = \quad \text{read-rn}(oplen, rn, \text{mc-rfile}(s)))$

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

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

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

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

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

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

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

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

```
; 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)
→ 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) ∧ ($l < r$))
→ 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) ∧ d5-7a4-5p (rn))
→ (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) ∧ ($l < r$) ∧ 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))

→ (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*)
∧ disjoint (sub (32, 52, read-sp (*s*)), 68, *x*, *k*))
→ (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*)
∧ qsort-sk (*s*, *s2*, *a*, *l*, *r*, *n*, *lst*))
→ 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*)))
→ 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*))
∧ (*oplen* ≤ 32)
∧ d2-7a2-5p (*rn*))
→ (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*))
∧ (*oplen* ≤ 32)
∧ d2-7a2-5p (*rn*))

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

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

THEOREM: qsort-sk-s-s5
(qsort-s4p (s, $s4$, a, l, r, n, qsort (l, r, lst)) $\wedge$ qsort-sk (s, $s4$, a, l, r, n, lst))
$\rightarrow$ qsort-sk (s, stepn ($s4$, qsort-3 (a, l, r, n, lst)), a, l, r, n, lst)

```
; some auxiliary lemmas.
```

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

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

THEOREM: qstack-la3
($l \not< r$) $\rightarrow$ (qstack (l, r, lst) $=$ 68)

THEOREM: qstack-0
(qstack (l, -1, lst) $=$ 68)
$\wedge$ (qstack (l, 0, lst) $=$ 68)
$\wedge$ (qstack (l, l, lst) $=$ 68)

THEOREM: qstack-la4
let $last$ **be** qlast (l, r, lst),
$lst1$ **be** qpart (l, r, lst)
in
qstack (l, r, lst) $\not<$ qstack (l, $last - 1$, $lst1$) **endlet**

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

THEOREM: qsort-s1-crock
(((($q1$ + 40) $\leq q$) $\wedge$ (68 $\leq q1$) $\wedge$ ($q <$ exp (2, 32)))
$\rightarrow$ ($q \not<$ ($q1$ + add (32,
4294967256,
add (32, neg (32, $q1 - 16$), $q - 16$))))

THEOREM: qstack-la1~ (($last$ $=$ qlast (l, r, lst)) $\wedge$ ($lst1$ $=$ qpart (l, r, lst)) $\wedge$ ($l < r$))
$\rightarrow$ (qstack (l, r, lst) $\not<$ (40 + qstack (l, $last - 1$, $lst1$)))

THEOREM: qsort-s1-s
let *lst1* **be** qpart (l, r, lst),
last **be** qlast (l, r, lst)
in
(qsort-s1p $(s,$ stepn $(s, k), a, l, r, n, lst1, last)$
$\wedge$ qsort-statep $(s, a, l, r, n, lst))$
$\rightarrow$ qsort-statep (stepn $(s, k), a, l,$ idifference $(last, 1), n, lst1)$ **endlet**

THEOREM: qsort-s1-crock1
$((68 \leq q1)$
$\wedge$ $((40 + q1) < 4294967296)$
$\wedge$ $((x + k) \leq 4294967296)$
$\wedge$ $(4294967272 \leq x))$
$\rightarrow$ disjoint (add $(32, 4294967256,$ neg $(32, q1 - 16)), q1, x, k)$

THEOREM: qsort-s1-crock2-crock
$((68 \leq q1) \wedge (q1 < 4294967256))$
$\rightarrow$ (add $(32, 4294967256,$ neg $(32, q1 - 16)) \not< 16)$

THEOREM: qsort-s1-crock2
$((68 \leq q1) \wedge ((40 + q1) < 4294967296) \wedge ((x + k) \leq 16))$
$\rightarrow$ disjoint (add $(32, 4294967256,$ 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
(qsort-s1p $(s, s1, a, l, r, n, lst1, last)$
$\wedge$ qsort-statep (s, a, l, r, n, lst)
$\wedge$ qsort-s5p $(s1,$
stepn $(s1, k)$,
a,
l,
idifference $(last, 1)$,
n,
qsort $(l, last - 1, lst1))$
$\wedge$ qsort-sk $(s1,$ stepn $(s1, k), a, l,$ idifference $(last, 1), n, lst1))$
$\rightarrow$ qsort-s2p $(s,$ stepn $(s1, k), a, l, r, n,$ qsort $(l, last - 1, lst1), last)$ **endlet**

EVENT: Disable qsort-s1-crock1.

EVENT: Disable qsort-s1-crock2.

THEOREM: qsort-s1-s2-mem
let *lst1* **be** qpart (l, r, lst),
last **be** qlast (l, r, lst)
in
(qsort-s1p $(s, s1, a, l, r, n, lst1, last)$
$\wedge$ qsort-statep (s, a, l, r, n, lst)
$\wedge$ qsort-sk $(s, s1, a, l, r, n, lst)$
$\wedge$ qsort-sk $(s1,$ stepn $(s1, k), a, l,$ idifference $(last, 1), n, lst1)$
$\wedge$ disjoint $(a, 4 * n, x, k1)$
$\wedge$ disjoint (sub (32, qstack $(l, r, lst) - 16$, read-sp (s)),
qstack (l, r, lst),
x,
$k1$))
$\rightarrow$ (read-mem $(x,$ mc-mem (stepn $(s1, k)), k1)$
$=$ read-mem $(x,$ mc-mem $(s), k1)$) **endlet**

EVENT: Disable qstack-la1~

THEOREM: qsort-sk-s-s2
let *lst1* **be** qpart (l, r, lst),
last **be** qlast (l, r, lst)
in
(qsort-s1p $(s, s1, a, l, r, n, lst1, last)$
$\wedge$ qsort-statep (s, a, l, r, n, lst)
$\wedge$ qsort-sk $(s, s1, a, l, r, n, lst)$
$\wedge$ qsort-sk $(s1,$ stepn $(s1, k), a, l,$ idifference $(last, 1), n, lst1))$
$\rightarrow$ qsort-sk $(s,$ stepn $(s1, k), a, l, r, n, lst)$ **endlet**

EVENT: Disable qsort-s1-s2-mem.

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

THEOREM: qsort-s3-crock
$(((q1 + 52) \leq q) \wedge (68 \leq q1) \wedge (q < \exp(2, 32)))$
$\rightarrow$ $(q \not< (q1 +$ add (32,
4294967244,
add (32, neg (32, $q1 - 16$), $q - 16$))))

THEOREM: qstack-la2~ $((last =$ qlast $(l, r, lst)) \wedge (lst1 =$ qpart $(l, r, lst)) \wedge (l < r))$
$\rightarrow$ (qstack (l, r, lst)
$\not<$ (52 + qstack $(1 + last, r,$ qsort $(l, last - 1, lst1))))$

THEOREM: qsort-s3-s-la
let *lst1* **be** qsort $(l,$ qlast $(l, r, lst) - 1,$ qpart $(l, r, lst))$,

```
    last  be  qlast (l, r, lst)
 in
 (qsort-s3p (s, stepn (s, k), a, l, r, n, lst1, last)
  ∧   qsort-statep (s, a, l, r, n, lst))
 →   qsort-statep (stepn (s, k), a, 1 + last, r, n, lst1) endlet
```

THEOREM: qsort-s3-s

```
 let lst1  be  qsort (l, qlast (l, r, lst) − 1, qpart (l, r, lst)),
     last  be  qlast (l, r, lst)
 in
 (qsort-s3p (s, stepn (stepn (stepn (s, k1), k2), k3), a, l, r, n, lst1, last)
  ∧   qsort-statep (s, a, l, r, n, lst))
 →   qsort-statep (stepn (stepn (stepn (s, k1), k2), k3),
                   a,
                   1 + last,
                   r,
                   n,
                   lst1) endlet
```

THEOREM: qsort-s3-crock1

```
 ((68 ≤ q1)
  ∧   ((52 + q1) < 4294967296)
  ∧   ((x + k) ≤ 4294967296)
  ∧   (4294967260 ≤ x))
 →   disjoint (add (32, 4294967244, neg (32, q1 − 16)), q1, x, k)
```

THEOREM: qsort-s3-crock2-crock

```
 ((68 ≤ q1) ∧ (q1 < 4294967244))
 →   (add (32, 4294967244, neg (32, q1 − 16)) ≮ 16)
```

THEOREM: qsort-s3-crock2

```
 ((68 ≤ q1) ∧ ((52 + q1) < 4294967296) ∧ ((x + k) ≤ 16))
 →   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, 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)
  ∧   qsort-statep (s, a, l, r, n, lst)
  ∧   qsort-s5p (s3,
                 stepn (s3, k),
```

```
                a,
                1 + last,
                r,
                n,
                qsort (1 + last, r, lst1))
 ∧   qsort-sk (s3, stepn (s3, k), a, 1 + last, r, n, lst1))
 →   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)
  ∧   qsort-statep (s, a, l, r, n, lst)
  ∧   qsort-sk (s, s3, a, l, r, n, lst)
  ∧   qsort-sk (s3, stepn (s3, k), a, 1 + last, r, n, lst1)
  ∧   disjoint (a, 4 ∗ n, x, k1)
  ∧   disjoint (sub (32, qstack (l, r, lst) − 16, read-sp (s)),
                qstack (l, r, lst),
                x,
                k1))
 →   (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)
  ∧   qsort-statep (s, a, l, r, n, lst)
  ∧   qsort-sk (s, s3, a, l, r, n, lst)
  ∧   qsort-sk (s3, stepn (s3, k), a, 1 + last, r, n, lst1))
 →   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, idifference (r, 1), n, lst) = qsort-t (a, l, $r - 1$, n, lst)

THEOREM: qsort–1
qsort (l, idifference (r, 1), lst) = qsort (l, $r - 1$, lst)

EVENT: Disable qsort-10.

EVENT: Disable qsort-5.

EVENT: Disable qsort-3.

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

THEOREM: qsort-correctness
let sn **be** stepn (s, qsort-t (a, l, r, n, lst)),
sp **be** sub (32, qstack (l, r, lst) − 16, read-sp (s))
in
qsort-statep (s, a, l, r, n, lst)
$\rightarrow$ ((mc-status (sn) = 'running)
$\wedge$ (mc-pc (sn) = rts-addr (s))
$\wedge$ (read-rn (32, 14, mc-rfile (sn))
= read-rn (32, 14, mc-rfile (s)))
$\wedge$ (read-rn (32, 15, mc-rfile (sn))
= add (32, read-rn (32, 15, mc-rfile (s)), 4))
$\wedge$ (((($oplen \leq$ 32) $\wedge$ d2-7a2-5p (rn))
$\rightarrow$ (read-rn ($oplen$, rn, mc-rfile (sn))
= read-rn ($oplen$, rn, mc-rfile (s))))
$\wedge$ ((disjoint (sp, qstack (l, r, lst), x, k)
$\wedge$ disjoint (a, 4 $*$ n, x, k))
$\rightarrow$ (read-mem (x, mc-mem (sn), k)
= read-mem (x, mc-mem (s), k)))
$\wedge$ mem-ilst (4, a, mc-mem (sn), n, 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
(fix $(i) \neq$ fix (j))
$\rightarrow$ (put-nth $(v1,\ i,$ put-nth $(v2,\ j,\ lst)) =$ put-nth $(v2,\ j,$ put-nth $(v1,\ i,\ lst)))$

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

THEOREM: swap-commute
$((i < l) \wedge (i < r) \wedge (j < l) \wedge (j < r))$
$\rightarrow$ (swap $(l,\ r,$ swap $(i,\ j,\ lst)) =$ swap $(i,\ j,$ swap $(l,\ r,\ lst)))$

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

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

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

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

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

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

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

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

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

THEOREM: sublsts-ileq-qpart-aux
$((l \leq last)$
$\wedge$ $(last < i)$
$\wedge$ $(last \leq r2)$
$\wedge$ $(l \leq l2)$
$\wedge$ $(l2 \leq r)$
$\wedge$ $(r2 \leq r)$)
$\rightarrow$ (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
$((l \leq l1) \wedge (r1 \leq r))$
$\rightarrow$ (sublst-ileq $(x, l, r,$ qsort $(l1, r1, lst)) =$ sublst-ileq $(x, l, r, lst))$

THEOREM: sublsts-ileq-qsort1
$((l \leq l2) \wedge (r2 \leq r))$
$\rightarrow$ (sublsts-ileq $(l, r,$ qsort $(l2, r2, lst), l1, r1, lst1)$
$=$ sublsts-ileq $(l, r, lst, l1, r1, lst1))$

THEOREM: sublsts-ileq-qsort2
$((l1 \leq l2) \wedge (r2 \leq r1))$
$\rightarrow$ (sublsts-ileq $(l, r, lst, l1, r1,$ qsort $(l2, r2, lst1))$
$=$ 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
$((last < i) \wedge (right < j))$
$\rightarrow$ (get-nth $(j,$ qpartx $(left, right, lst, last, i)) =$ get-nth $(j, lst))$

THEOREM: qpartx-get-2
$((last < i) \wedge (j \leq last))$
$\rightarrow$ (get-nth $(j,$ qpartx $(left, right, lst, last, i)) =$ get-nth $(j, lst))$

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

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

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

THEOREM: open-sublst-ileq-la1
(open-sublst-ileq $(x,\ last,\ i,\ lst) \wedge (last < j) \wedge (j < i)$)
$\rightarrow$ ($\neg$ ilessp (get-nth $(j,\ lst)$, x))

THEOREM: open-sublst-ileq-la2
open-sublst-ileq $(x,\ last,\ 1 + i,\ lst)$
$=$ **if** ileq $(x$, get-nth $(i,\ lst))$ **then** open-sublst-ileq $(x,\ last,\ i,\ lst)$
else $(1 + last) \not< (1 + i)$ **endif**

THEOREM: open-sublst-ileq-la3
$(j \leq l)$
$\rightarrow$ (open-sublst-ileq $(x,\ l,\ r,$ put-nth $(v,\ j,\ lst))$
$=$ open-sublst-ileq $(x,\ l,\ r,\ lst)$)

THEOREM: open-sublst-ileq-la4
$(r \leq j)$
$\rightarrow$ (open-sublst-ileq $(x,\ l,\ r,$ put-nth $(v,\ j,\ lst))$
$=$ open-sublst-ileq $(x,\ l,\ r,\ lst)$)

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

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

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

THEOREM: qpart-ilessp-la1-la
$((l \leq j) \wedge (j <$ qlast-aux $(l,\ r,\ lst,\ l,\ 1 + l)))$
$\rightarrow$ ($\neg$ ilessp (get-nth $(l,\ lst)$, get-nth $(j,$ qpart-aux $(l,\ r,\ lst,\ l,\ 1 + l))))$

THEOREM: qpart-ilessp-la1
$((l \leq j) \wedge (j \leq$ (qlast-aux $(l,\ r,\ lst,\ l,\ 1 + l) - 1)))$
$\rightarrow$ ($\neg$ ilessp (get-nth $(l,\ lst)$, get-nth $(j,$ qpart-aux $(l,\ r,\ lst,\ l,\ 1 + l))))$

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

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

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

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

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

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

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

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

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

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

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

EVENT: Disable qpart-aux-qpartx.

THEOREM: ilessp-trans
(ilessp $(a,\ b) \wedge$ ileq $(b,\ c)) \rightarrow$ ilessp $(a,\ c)$

THEOREM: qpart-ilessp
$((l \leq i)$
$\wedge\quad (i \leq$ qlast $(l,\ r,\ lst))$
$\wedge\quad ($qlast $(l,\ r,\ lst) \leq j)$
$\wedge\quad (j \leq r))$
$\rightarrow\quad (\neg$ ilessp (get-nth $(j,$ qpart $(l,\ r,\ lst))$, get-nth $(i,$ qpart $(l,\ r,\ lst))))$

THEOREM: qsort-get3
get-nth $(j,$ qsort $(1 + last,\ r,$ qsort $(l,\ last - 1,\ lst)))$
$=\quad$ **if** $j \leq (last - 1)$ **then** get-nth $(j,$ qsort $(l,\ last - 1,\ lst))$
else get-nth $(j,$ qsort $(1 + last,\ r,\ lst))$ **endif**

THEOREM: sublsts-ileq-la2
(sublsts-ileq $(l,\ r,\ lst,\ l1,\ r1,\ lst1)$
$\wedge\quad (l \leq i)$
$\wedge\quad (i \leq r)$
$\wedge\quad (l1 \leq j)$
$\wedge\quad (j \leq r1))$
$\rightarrow\quad (\neg$ ilessp (get-nth $(j,\ lst1)$, get-nth $(i,\ lst)))$

THEOREM: qpart-ilessp-closed-1
(qlast $(l,\ r,\ lst) \leq last)$
$\rightarrow\quad$ sublst-ileq (get-nth (qlast $(l,\ r,\ lst)$, qpart $(l,\ r,\ lst))$,
$last$,
r,
qpart $(l,\ r,\ lst))$

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

THEOREM: qpart-ilessp-closed-2
let $lst1$ **be** qpart $(l,\ r,\ lst)$
in
((qlast $(l,\ r,\ lst) \leq last)$
$\wedge\quad (l \leq j)$

$\wedge \quad (j \leq (\text{qlast}\,(l,\ r,\ lst) - 1)))$
$\rightarrow \quad \text{sublst-ileq}\,(\text{get-nth}\,(j,\ lst1),\ last,\ r,\ lst1)$ **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
$(i \notin \mathbf{N})$
$\rightarrow \quad ((\text{swap}\,(i,\ j,\ lst) = \text{swap}\,(\mathtt{0},\ j,\ lst))$
$\quad \wedge \quad (\text{swap}\,(j,\ i,\ lst) = \text{swap}\,(j,\ \mathtt{0},\ lst)))$

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

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

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

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

THEOREM: count-lst-qsort-la

$$
\begin{array}{ll}
((l \leq l1) \wedge (r1 \leq r)) & \\
\rightarrow & (\text{count-lst}\,(x,\ l,\ r,\ \text{qsort}\,(l1,\ r1,\ lst)) = \text{count-lst}\,(x,\ l,\ r,\ lst))
\end{array}
$$

THEOREM: count-lst-qsort

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

Index

add, 8–13, 20–23, 25
add1-int-rangep, 15
add1-int-rangepxx, 16

count-lst, 32–34
count-lst-0, 33
count-lst-eq, 33
count-lst-put-1, 33
count-lst-qpart-aux, 34
count-lst-qsort, 34
count-lst-qsort-la, 34
count-lst-swap, 33
count-lst-swap-1, 33
count-lst-swap-rec, 33
count-swapii, 33
count-transwap, 33
count-transwap-0, 33

d2-7a2-5p, 14, 19, 25
d5-7a4-5p, 15–20
disjoint, 8–16, 18–25

equal*, 9–13
evenp, 8–13
exp, 8, 20, 22

forall, 13

get-nth, 4–6, 16, 17, 26, 28–32
get-swap-1, 30

idifference, 7, 11, 21, 22, 25
ileq, 26, 28, 29, 31, 32
ilessp, 4–6, 16, 17, 26, 28–32
ilessp-lessp, 4
ilessp-trans, 31
int-rangep, 9–12, 15, 16
int-rangep-plus-1, 15
iread-mem, 8–12

linked-a6, 9–13
linked-rts-addr, 9–13

lst-eq, 32, 33

mc-mem, 8–16, 18–20, 22, 24, 25
mc-pc, 8–13, 25
mc-rfile, 9–20, 25
mc-status, 8–13, 25
mcode-addrp, 8–13
mean-bounds, 15
mem-ilst, 8–13, 25
movem-saved, 9–13

nat-to-int, 9–12, 15
neg, 20–23

open-sublst-ileq, 28, 29
open-sublst-ileq-la0, 29
open-sublst-ileq-la1, 29
open-sublst-ileq-la2, 29
open-sublst-ileq-la3, 29
open-sublst-ileq-la4, 29
open-sublst-ileq-la5, 29
open-sublst-ileq-la6, 29
orderedp1, 32

put-commute, 26
put-nth, 26, 29, 33

qlast, 5–8, 17–24, 30–32
qlast-0, 20
qlast-aux, 5, 17, 20, 28–30
qlast-aux-0, 20
qlast-aux-lb, 5
qlast-aux-swap, 30
qlast-aux-ub, 5
qlast-lb, 5
qlast-swap, 30
qlast-ub, 5
qpart, 5–8, 17–24, 30–32
qpart-aux, 4, 5, 17, 27–30, 34
qpart-aux-ct, 17
qpart-aux-mem, 18
qpart-aux-qpartx, 28

qpart-aux-rfile, 17
qpart-aux-swap, 30
qpart-aux-t, 6, 17, 18
qpart-equal, 30
qpart-ilessp, 31
qpart-ilessp-closed-1, 31
qpart-ilessp-closed-2, 31
qpart-ilessp-closed-3, 32
qpart-ilessp-la1, 29
qpart-ilessp-la1-la, 29
qpart-ilessp-la2, 30
qpart-induct, 17
qpart-swap, 30
qpart-t, 6, 7, 17, 18
qpartx, 28–30
qpartx-get-1, 28
qpartx-get-2, 28
qpartx-ilessp-1, 28
qpartx-ilessp-2, 29
qsort, 5–8, 18–25, 28, 30–32, 34
qsort–1, 25
qsort-10, 6, 14
qsort-3, 6, 19, 20
qsort-5, 6, 7, 18, 19
qsort-base, 14
qsort-base-mem, 14
qsort-base-mem-sk, 14
qsort-base-rfile, 14
qsort-code, 4, 8–13
qsort-correctness, 25
qsort-correctness-la, 25
qsort-get-1, 30
qsort-get-2, 30
qsort-get3, 31
qsort-ilessp-1, 31
qsort-ilessp-2, 32
qsort-induct, 6, 7
qsort-ordered, 32
qsort-orderedp1, 32
qsort-orderedp1-la, 32
qsort-qpartx, 30
qsort-s-s0, 15
qsort-s-s0-mem, 15
qsort-s-s0-rfile, 15
qsort-s-s1, 17
qsort-s-s1-rfile, 17
qsort-s0-s0-1, 16
qsort-s0-s0-2, 16
qsort-s0-s0-mem-1, 16
qsort-s0-s0-mem-2, 16
qsort-s0-s0-rfile-1, 16
qsort-s0-s0-rfile-2, 16
qsort-s0-s1, 15
qsort-s0-s1-mem, 15
qsort-s0-s1-rfile, 15
qsort-s0p, 9, 15–18
qsort-s1-crock, 20
qsort-s1-crock1, 21
qsort-s1-crock2, 21
qsort-s1-crock2-crock, 21
qsort-s1-s, 21
qsort-s1-s2, 21
qsort-s1-s2-mem, 22
qsort-s1p, 10, 15, 17, 21, 22
qsort-s2-s3, 18
qsort-s2-s3-mem, 19
qsort-s2-s3-rfile, 18
qsort-s2-s3-rfile-la, 18
qsort-s2p, 11, 18, 19, 21
qsort-s3-crock, 22
qsort-s3-crock1, 23
qsort-s3-crock2, 23
qsort-s3-crock2-crock, 23
qsort-s3-s, 23
qsort-s3-s-la, 22
qsort-s3-s4, 23
qsort-s3-s4-mem, 24
qsort-s3p, 12, 18, 23, 24
qsort-s4-s5, 19
qsort-s4-s5-mem, 20
qsort-s4-s5-rfile, 19
qsort-s4-s5-rfile-la, 19
qsort-s4p, 12, 13, 19, 20, 24
qsort-s5p, 13, 14, 19, 21, 24, 25
qsort-sk, 13, 14, 18–22, 24, 25
qsort-sk-1, 14
qsort-sk-2, 14
qsort-sk-s-s1, 18

qsort-sk-s-s2, 22
qsort-sk-s-s3, 19
qsort-sk-s-s4, 24
qsort-sk-s-s5, 20
qsort-sp, 9, 14, 15, 17, 18
qsort-statep, 8, 9, 21–25
qsort-statep-sp, 9
qsort-swap, 30
qsort-t, 6, 7, 25
qsort-t–1, 25
qstack, 7, 8, 13, 14, 20, 22, 24, 25
qstack-0, 20
qstack-la0, 7
qstack-la1, 7
qstack-la1-, 20
qstack-la2, 7
qstack-la2-, 22
qstack-la3, 20
qstack-la4, 20
qstack-ubound, 8
qstack-ubound-la-1, 8
qstack-ubound-la-2, 8

ram-addrp, 8–13
read-an, 9–13
read-dn, 13
read-mem, 8–10, 12–16, 18–20, 22, 24, 25
read-rn, 9–20, 25
read-sp, 8–15, 18–20, 22, 24, 25
readm-mem, 14
readm-rn, 9–13
rom-addrp, 8–13
rts-addr, 9–13, 25

splus, 6, 7
stepn, 7, 14–25
sub, 8–15, 18–20, 22, 24, 25
sublst-ileq, 26–28, 31, 32
sublst-ileq-lemma, 26
sublst-ileq-put, 26
sublst-ileq-qpart-aux, 27
sublst-ileq-qsort, 28
sublst-ileq-swap, 27
sublst-ileq-swap-la, 26
sublst-ileq-swap-swap, 27
sublsts-ileq, 26–28, 31, 32
sublsts-ileq-la1, 26
sublsts-ileq-la2, 31
sublsts-ileq-put, 26
sublsts-ileq-qpart-aux, 27
sublsts-ileq-qsort1, 28
sublsts-ileq-qsort2, 28
sublsts-ileq-swap, 27
sublsts-ileq-swap-la, 27
sublsts-ileq-swap-swap, 27
swap, 4–6, 15–17, 26–30, 32, 33
swap-0, 33
swap-commute, 26
swap-induct, 33
swap-put-commute, 26
swap-rec-la, 33

transwap, 32, 33

uint-rangep, 8–12