```
#|

 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.

|#

;         Proof of the Correctness of a Binary Search Program
;
```

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

```
#|
The following C function BSEARCH determines if a given value x occurs
in the sorted array a.

/* from K&R */
/* bsearch:  find x in a[0] <= a[1] <= ... <= a[n-1] */
int bsearch (int x, int a[], int n)
{
  int low, high, mid;

  low = 0;
  high = n;
  while (low < high) {
    mid = (low + high) / 2;
```

```
    if (x < a[mid])
      high = mid;
    else if (x > a[mid])
      low = mid + 1;
    else return mid;
  }
  return -1;
}
```

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

```
0x22f0 <bsearch>:       linkw a6,#0
0x22f4 <bsearch+4>:     moveml d2-d3,sp@-
0x22f8 <bsearch+8>:     movel a6@(8),d3
0x22fc <bsearch+12>:    moveal a6@(12),a0
0x2300 <bsearch+16>:    clrl d1
0x2302 <bsearch+18>:    movel a6@(16),d2
0x2306 <bsearch+22>:    cmpl d1,d2
0x2308 <bsearch+24>:    ble 0x232a <bsearch+58>
0x230a <bsearch+26>:    movel d1,d0
0x230c <bsearch+28>:    addl d2,d0
0x230e <bsearch+30>:    bpl 0x2312 <bsearch+34>
0x2310 <bsearch+32>:    addql #1,d0
0x2312 <bsearch+34>:    asrl #1,d0
0x2314 <bsearch+36>:    cmpl 0(a0)[d0.l*4],d3
0x2318 <bsearch+40>:    bge 0x231e <bsearch+46>
0x231a <bsearch+42>:    movel d0,d2
0x231c <bsearch+44>:    bra 0x2306 <bsearch+22>
0x231e <bsearch+46>:    cmpl 0(a0)[d0.l*4],d3
0x2322 <bsearch+50>:    ble 0x232c <bsearch+60>
0x2324 <bsearch+52>:    movel d0,d1
0x2326 <bsearch+54>:    addql #1,d1
0x2328 <bsearch+56>:    bra 0x2306 <bsearch+22>
0x232a <bsearch+58>:    movel #-1,d0
0x232c <bsearch+60>:    moveml a6@(-8),d2-d3
0x2332 <bsearch+66>:    unlk a6
0x2334 <bsearch+68>:    rts
```

The machine code of the above program is:

```
<bsearch>:       0x4e56  0x0000  0x48e7  0x3000  0x262e  0x0008  0x206e  0x000c
<bsearch+16>:    0x4281  0x242e  0x0010  0xb481  0x6f20  0x2001  0xd082  0x6a02
<bsearch+32>:    0x5280  0xe280  0xb6b0  0x0c00  0x6c04  0x2400  0x60e8  0xb6b0
```

```
<bsearch+48>:    0x0c00  0x6f08  0x2200  0x5281  0x60dc  0x70ff  0x4cee  0x000c
<bsearch+64>:    0xfff8  0x4e5e  0x4e75

In the Logic, this is:

'(78      86      0       0       72      231     48      0
  38      46      0       8       32      110     0       12
  66      129     36      46      0       16      180     129
  111     32      32      1       208     130     106     2
  82      128     226     128     182     176     12      0
  108     4       36      0       96      232     182     176
  12      0       111     8       34      0       82      129
  96      220     112     255     76      238     0       12
  255     248     78      94      78      117)

|#

; in the logic, the above program is defined by (bsearch-code).
```

DEFINITION:
BSEARCH-CODE
= `'(78 86 0 0 72 231 48 0 38 46 0 8 32 110 0 12 66 129 36 46 0 16 180 129 111 32 32 1 208 130 106 2 82 128 226 128 182 176 12 0 108 4 36 0 96 232 182 176 12 0 111 8 34 0 82 129 96 220 112 255 76 238 0 12 255 248 78 94 78 117)`

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

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

EVENT: Disable ilessp.

```
; bsearch1 is a function in the logic to simulate the loop of the
; above code.
```

DEFINITION:
bsearch1 $(x,\, lst,\, i,\, j)$
= **let** k **be** $(i + j) \div 2$
in
if $i < j$
then if ilessp $(x,$ get-nth $(k,\, lst))$ **then** bsearch1 $(x,\, lst,\, i,\, k)$

```
        elseif ilessp (get-nth (k, lst), x)
        then bsearch1 (x, lst, 1 + k, j)
        else k endif
    else -1 endif endlet

; bsearch is a function in the logic to simulate the above code.
```

DEFINITION: bsearch (x, n, lst) = bsearch1 $(x, lst, 0, n)$

```
; the computation time of the loop.
```

DEFINITION:
bsearch1-t (x, lst, i, j)
= **let** k **be** $(i + j) \div 2$
in
if $i < j$
then if ilessp $(x,$ get-nth $(k, lst))$
then splus $(10,$ bsearch1-t $(x, lst, i, k))$
elseif ilessp (get-nth $(k, lst), x)$
then splus $(13,$ bsearch1-t $(x, lst, 1 + k, j))$
else 13 **endif**
else 6 **endif endlet**

```
; the computation time of the code.
```

DEFINITION:
bsearch-t (x, n, lst) = splus $(6,$ bsearch1-t $(x, lst, 0, n))$

```
; an induction hint.
```

DEFINITION:
bsearch-induct (s, x, lst, i, j)
= **let** k **be** $(i + j) \div 2$
in
if $i < j$
then if ilessp $(x,$ get-nth $(k, lst))$
then bsearch-induct (stepn $(s, 10), x, lst, i, k)$
elseif ilessp (get-nth $(k, lst), x)$
then bsearch-induct (stepn $(s, 13), x, lst, 1 + k, j)$
else t endif
else t endif endlet

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

DEFINITION:
bsearch-statep (s, x, a, n, lst)
= ((mc-status (s) = '`running`)
∧ evenp (mc-pc (s))
∧ rom-addrp (mc-pc (s), mc-mem (s), 70)
∧ mcode-addrp (mc-pc (s), mc-mem (s), BSEARCH-CODE)
∧ ram-addrp (sub (32, 12, read-sp (s)), mc-mem (s), 28)
∧ ram-addrp (a, mc-mem (s), 4 ∗ n)
∧ mem-ilst (4, a, mc-mem (s), n, lst)
∧ disjoint (sub (32, 12, read-sp (s)), 28, a, 4 ∗ n)
∧ (a = read-mem (add (32, read-sp (s), 8), mc-mem (s), 4))
∧ (n = iread-mem (add (32, read-sp (s), 12), mc-mem (s), 4))
∧ (x = iread-mem (add (32, read-sp (s), 4), mc-mem (s), 4))
∧ int-rangep (2 ∗ n, 32)
∧ (n ∈ **N**))

```
; the conditions of an intermediate state.
```

DEFINITION:
bsearch-s0p (s, x, a, n, lst, i, j)
= ((mc-status (s) = '`running`)
∧ evenp (mc-pc (s))
∧ rom-addrp (sub (32, 22, mc-pc (s)), mc-mem (s), 70)
∧ mcode-addrp (sub (32, 22, mc-pc (s)), mc-mem (s), BSEARCH-CODE)
∧ ram-addrp (sub (32, 8, read-an (32, 6, s)), mc-mem (s), 28)
∧ ram-addrp (a, mc-mem (s), 4 ∗ n)
∧ mem-ilst (4, a, mc-mem (s), n, lst)
∧ disjoint (sub (32, 8, read-an (32, 6, s)), 28, a, 4 ∗ n)
∧ (a = read-an (32, 0, s))
∧ (i = nat-to-int (read-dn (32, 1, s), 32))
∧ (j = nat-to-int (read-dn (32, 2, s), 32))
∧ (x = nat-to-int (read-dn (32, 3, s), 32))
∧ int-rangep (2 ∗ j, 32)
∧ (i ∈ **N**)
∧ (j ∈ **N**)
∧ (n ∈ **N**)
∧ (i ≤ n)
∧ (j ≤ n))

```
; the initial segment.  From the initial state to s0.
```

THEOREM: bsearch-s-s0p
bsearch-statep (s, x, a, n, lst) → bsearch-s0p (stepn (s, 6), x, a, n, lst, 0, n)

THEOREM: bsearch-s-s0

bsearch-statep (s, x, a, n, lst)
→ ((linked-rts-addr (stepn (s, 6)) = rts-addr (s))
∧ (linked-a6 (stepn (s, 6)) = read-an (32, 6, s))
∧ (read-rn (32, 14, mc-rfile (stepn (s, 6)))
= sub (32, 4, read-sp (s)))
∧ (movem-saved (stepn (s, 6), 4, 8, 2)
= readm-rn (32, '(2 3), mc-rfile (s))))

THEOREM: bsearch-s-s0-rfile
(bsearch-statep (s, x, a, n, lst) ∧ d4-7a2-5p (rn))
→ (read-rn ($oplen$, rn, mc-rfile (stepn (s, 6)))
= read-rn ($oplen$, rn, mc-rfile (s)))

THEOREM: bsearch-s-s0-mem
(bsearch-statep (s, x, a, n, lst) ∧ disjoint (x, k, sub (32, 12, read-sp (s)), 28))
→ (read-mem (x, mc-mem (stepn (s, 6)), k) = read-mem (x, mc-mem (s), k))

```
; from s0 to s0 (induction case), from s0 to exit (base case).
; base case: s0 --> sn.
```

THEOREM: bsearch-s0-sn-base1
(bsearch-s0p (s, x, a, n, lst, i, j) ∧ ($i \not< j$))
→ ((mc-status (stepn (s, 6)) = 'running)
∧ (mc-pc (stepn (s, 6)) = linked-rts-addr (s))
∧ (iread-dn (32, 0, stepn (s, 6)) = -1)
∧ (read-rn (32, 14, mc-rfile (stepn (s, 6))) = linked-a6 (s))
∧ (read-rn (32, 15, mc-rfile (stepn (s, 6)))
= add (32, read-an (32, 6, s), 8))
∧ (read-mem (x, mc-mem (stepn (s, 6)), l)
= read-mem (x, mc-mem (s), l)))

THEOREM: bsearch-s0-sn-rfile-base1
(bsearch-s0p (s, x, a, n, lst, i, j)
∧ ($i \not< j$)
∧ d2-7a2-5p (rn)
∧ ($oplen \leq$ 32))
→ (read-rn ($oplen$, rn, mc-rfile (stepn (s, 6)))
= **if** d4-7a2-5p (rn) **then** read-rn ($oplen$, rn, mc-rfile (s))
else get-vlst ($oplen$, 0, rn, '(2 3), movem-saved (s, 4, 8, 2)) **endif**)

EVENT: Enable iplus.

EVENT: Enable iquotient.

EVENT: Disable quotient.

EVENT: Disable remainder.

EVENT: Disable times.

THEOREM: bsearch-crock
(int-rangep (2 * j, n) ∧ (i < j)) → int-rangep (i + j, n)

THEOREM: bsearch-s0-sn-base2
let k **be** (i + j) ÷ 2
in
(bsearch-s0p (s, x, a, n, *lst*, i, j)
∧ (i < j)
∧ (¬ ilessp (x, get-nth (k, *lst*)))
∧ (¬ ilessp (get-nth (k, *lst*), x)))
→ ((mc-status (stepn (s, 13)) = 'running)
∧ (mc-pc (stepn (s, 13)) = linked-rts-addr (s))
∧ (iread-dn (32, 0, stepn (s, 13)) = ((i + j) ÷ 2))
∧ (read-rn (32, 14, mc-rfile (stepn (s, 13)))
= linked-a6 (s))
∧ (read-rn (32, 15, mc-rfile (stepn (s, 13)))
= add (32, read-an (32, 6, s), 8))
∧ (read-mem (x, mc-mem (stepn (s, 13)), l)
= read-mem (x, mc-mem (s), l))) **endlet**

THEOREM: bsearch1-s0-sn-rfile-base2
let k **be** (i + j) ÷ 2
in
(bsearch-s0p (s, x, a, n, *lst*, i, j)
∧ (i < j)
∧ (¬ ilessp (x, get-nth (k, *lst*)))
∧ (¬ ilessp (get-nth (k, *lst*), x))
∧ d2-7a2-5p (*rn*)
∧ (*oplen* ≤ 32))
→ (read-rn (*oplen*, *rn*, mc-rfile (stepn (s, 13)))
= **if** d4-7a2-5p (*rn*) **then** read-rn (*oplen*, *rn*, mc-rfile (s))
else get-vlst (*oplen*,
0,
rn,
'(2 3),
movem-saved (s, 4, 8, 2)) **endif**) **endlet**

```
; from s0 to s0 (induction case).
```

THEOREM: bsearch-s0-s0-1

let k **be** $(i + j) \div 2$
in
(bsearch-s0p (s, x, a, n, *lst*, i, j)
$\wedge$ $(i < j)$
$\wedge$ ilessp (x, get-nth (k, *lst*)))
$\rightarrow$ (bsearch-s0p (stepn (s, 10), x, a, n, *lst*, i, k)
$\wedge$ (read-rn (*oplen*, 14, mc-rfile (stepn (s, 10)))
$=$ read-rn (*oplen*, 14, mc-rfile (s)))
$\wedge$ (linked-a6 (stepn (s, 10)) = linked-a6 (s))
$\wedge$ (linked-rts-addr (stepn (s, 10)) = linked-rts-addr (s))
$\wedge$ (movem-saved (stepn (s, 10), 4, 8, 2)
$=$ movem-saved (s, 4, 8, 2))
$\wedge$ (read-mem (x, mc-mem (stepn (s, 10)), l)
$=$ read-mem (x, mc-mem (s), l))) **endlet**

THEOREM: bsearch-s0-s0-rfile1
let k **be** $(i + j) \div 2$
in
(bsearch-s0p (s, x, a, n, *lst*, i, j)
$\wedge$ $(i < j)$
$\wedge$ ilessp (x, get-nth (k, *lst*))
$\wedge$ d4-7a2-5p (*rn*))
$\rightarrow$ (read-rn (*oplen*, *rn*, mc-rfile (stepn (s, 10)))
$=$ read-rn (*oplen*, *rn*, mc-rfile (s))) **endlet**

THEOREM: bsearch-s0-s0-2
let k **be** $(i + j) \div 2$
in
(bsearch-s0p (s, x, a, n, *lst*, i, j)
$\wedge$ $(i < j)$
$\wedge$ $(\neg$ ilessp (x, get-nth (k, *lst*)))
$\wedge$ ilessp (get-nth (k, *lst*), x))
$\rightarrow$ (bsearch-s0p (stepn (s, 13), x, a, n, *lst*, $1 + k$, j)
$\wedge$ (read-rn (*oplen*, 14, mc-rfile (stepn (s, 13)))
$=$ read-rn (*oplen*, 14, mc-rfile (s)))
$\wedge$ (linked-a6 (stepn (s, 13)) = linked-a6 (s))
$\wedge$ (linked-rts-addr (stepn (s, 13)) = linked-rts-addr (s))
$\wedge$ (movem-saved (stepn (s, 13), 4, 8, 2)
$=$ movem-saved (s, 4, 8, 2))
$\wedge$ (read-mem (x, mc-mem (stepn (s, 13)), l)
$=$ read-mem (x, mc-mem (s), l))) **endlet**

THEOREM: bsearch1-s0-s0-rfile2
let k **be** $(i + j) \div 2$
in

(bsearch-s0p (s, x, a, n, lst, i, j)
 $\wedge$ $(i < j)$
 $\wedge$ $(\neg$ ilessp (x, get-nth (k, lst)))
 $\wedge$ ilessp (get-nth (k, lst), x)
 $\wedge$ d4-7a2-5p (rn))
$\rightarrow$ (read-rn ($oplen$, rn, mc-rfile (stepn (s, 13)))
 $=$ read-rn ($oplen$, rn, mc-rfile (s))) **endlet**

EVENT: Disable bsearch-statep.

EVENT: Disable bsearch-s0p.

```
; put together. s0 --> sn
```

THEOREM: bsearch-s0-sn
let sn **be** stepn (s, bsearch1-t (x, lst, i, j))
in
bsearch-s0p (s, x, a, n, lst, i, j)
$\rightarrow$ ((mc-status (sn) = 'running)
 $\wedge$ (mc-pc (sn) = linked-rts-addr (s))
 $\wedge$ (iread-dn (32, 0, sn) = bsearch1 (x, lst, i, j))
 $\wedge$ (read-rn (32, 14, mc-rfile (sn)) = linked-a6 (s))
 $\wedge$ (read-rn (32, 15, mc-rfile (sn))
 $=$ add (32, read-an (32, 6, s), 8))
 $\wedge$ (read-mem (x, mc-mem (sn), k) = read-mem (x, mc-mem (s), k))) **endlet**

THEOREM: bsearch-s0-sn-rfile
(bsearch-s0p (s, x, a, n, lst, i, j) $\wedge$ d2-7a2-5p (rn) $\wedge$ ($oplen \leq 32$))
$\rightarrow$ (read-rn ($oplen$, rn, mc-rfile (stepn (s, bsearch1-t (x, lst, i, j))))
 $=$ **if** d4-7a2-5p (rn) **then** read-rn ($oplen$, rn, mc-rfile (s))
 else get-vlst ($oplen$, 0, rn, '(2 3), movem-saved (s, 4, 8, 2)) **endif**)

```
; the correctness statement of this BSEARCH program.
; after an execution of this program,  the machine state satisfies:
;  0. normal exit.
;  1. the program counter is returned to the next instruction of the caller.
;  2. the result -- (bsearch x n lst), is stored in the register D0.
;  3. a6, used by LINK, is restored to its original value.
;  4. a7, the stack pointer, is updated correctly to pop off one frame.
;  5. the registers d2-d7 and a2-a5 maintain their original values.
;  6. the memory is only locally changed wrt this program.
```

THEOREM: bsearch-correctness
let sn **be** stepn (s, bsearch-t (x, n, lst))

in
bsearch-statep (s, x, a, n, lst)
→ ((mc-status (sn) = `'running`)
∧ (mc-pc (sn) = rts-addr (s))
∧ (read-rn (`32`, `14`, mc-rfile (sn))
= read-rn (`32`, `14`, mc-rfile (s)))
∧ (read-rn (`32`, `15`, mc-rfile (sn))
= add (`32`, read-sp (s), `4`))
∧ ((d2-7a2-5p (rn) ∧ ($oplen$ ≤ `32`))
→ (read-rn ($oplen$, rn, mc-rfile (sn))
= read-rn ($oplen$, rn, mc-rfile (s))))
∧ (disjoint (x, k, sub (`32`, `12`, read-sp (s)), `28`)
→ (read-mem (x, mc-mem (sn), k)
= read-mem (x, mc-mem (s), k)))
∧ (iread-dn (`32`, `0`, sn) = bsearch (x, n, lst))) **endlet**

EVENT: Disable bsearch-t.

```
; in the logic, bsearch has these properties:
;      1. if it returns a nonnegative integer i, then lst[i] = x.
;      2. if it returns -1, then x is not in lst.
```

DEFINITION:
member1 (x, lst, i, j)
= **if** $i < j$
then if x = get-nth (i, lst) **then t**
else member1 (x, lst, 1 + i, j) **endif**
else f endif

DEFINITION:
orderedp (lst)
= **if** $lst \simeq$ **nil then t**
elseif cdr (lst) ≃ **nil then t**
else ileq (car (lst), cadr (lst)) ∧ orderedp (cdr (lst)) **endif**

THEOREM: leq-trans
((¬ ilessp (x, y)) ∧ (¬ ilessp (y, z))) → (¬ ilessp (x, z))

THEOREM: int-equal
(integerp (x) ∧ integerp (y) ∧ (¬ ilessp (x, y)) ∧ (¬ ilessp (y, x)))
→ (($x = y$) = **t**)

THEOREM: orderedp-ordered
(orderedp (lst) ∧ ($i \le j$) ∧ ($j <$ len (lst)))
→ (ilessp (get-nth (j, lst), get-nth (i, lst)) = **f**)

THEOREM: bsearch1-found
((bsearch1 (x, lst, i, j) $\neq$ -1) $\wedge$ lst-integerp (lst) $\wedge$ integerp (x))
$\rightarrow$ (get-nth (bsearch1 (x, lst, i, j), lst) = x)

THEOREM: bsearch1-not-found-1
(orderedp (lst)
$\wedge$ ilessp (get-nth (k, lst), x)
$\wedge$ ($i \leq k$)
$\wedge$ ($k < j$)
$\wedge$ ($j \leq$ len (lst)))
$\rightarrow$ (member1 (x, lst, i, j) = member1 (x, lst, $1 + k$, j))

THEOREM: bsearch1-not-found-2-lemma
(orderedp (lst)
$\wedge$ ilessp (x, get-nth (k, lst))
$\wedge$ ($k \leq i$)
$\wedge$ ($j <$ len (lst)))
$\rightarrow$ ($\neg$ member1 (x, lst, i, j))

THEOREM: bsearch1-not-found-2
(orderedp (lst)
$\wedge$ ilessp (x, get-nth (k, lst))
$\wedge$ ($k < j$)
$\wedge$ ($j \leq$ len (lst)))
$\rightarrow$ (member1 (x, lst, i, j) = member1 (x, lst, i, k))

EVENT: Disable bsearch1-not-found-2-lemma.

THEOREM: bsearch1-not-found
((bsearch1 (x, lst, i, j) = -1)
$\wedge$ orderedp (lst)
$\wedge$ lst-integerp (lst)
$\wedge$ integerp (x)
$\wedge$ ($j \leq$ len (lst)))
$\rightarrow$ ($\neg$ member1 (x, lst, i, j))

DEFINITION:
member2 (x, lst, i, j)
= **if** $i < j$
then if x = get-nth (i, lst) **then t**
else member2 (x, cdr (lst), i, $j - 1$) **endif**
else f endif

THEOREM: member2-member
member2 (x, lst, 0, len (lst)) = ($x \in lst$)

THEOREM: member2-lemma
$(x \neq \text{get-nth}\,(i,\ lst)) \rightarrow (\text{member2}\,(x,\ lst,\ 1 + i,\ j) = \text{member2}\,(x,\ lst,\ i,\ j))$

THEOREM: member1-member2
$\text{member1}\,(x,\ lst,\ i,\ j) = \text{member2}\,(x,\ lst,\ i,\ j)$

THEOREM: bsearch-found
$((\text{bsearch}\,(x,\ n,\ lst) \neq \texttt{-1}) \wedge \text{lst-integerp}\,(lst) \wedge \text{integerp}\,(x))$
$\rightarrow \quad (\text{get-nth}\,(\text{bsearch}\,(x,\ n,\ lst),\ lst) = x)$

THEOREM: bsearch-not-found
$((\text{bsearch}\,(x,\ \text{len}\,(lst),\ lst) = \texttt{-1})$
$\wedge \quad \text{orderedp}\,(lst)$
$\wedge \quad \text{lst-integerp}\,(lst)$
$\wedge \quad \text{integerp}\,(x))$
$\rightarrow \quad (x \notin lst)$

```
; an upper bound.
```

THEOREM: bsearch1-t-0
$\text{bsearch1-t}\,(x,\ lst,\ i,\ i) = \texttt{6}$

THEOREM: bsearch1-t-crock
$((i < j)$
$\wedge \quad ((\texttt{26} + (\texttt{13} * \log\,(\texttt{2},\ (j - \texttt{1}) - ((i + j) \div \texttt{2}))))$
$\not< \quad \text{bsearch1-t}\,(x,\ lst,\ 1 + ((i + j) \div \texttt{2}),\ j)))$
$\rightarrow \quad (((\texttt{26} + (\texttt{13} * \log\,(\texttt{2},\ j - i)))$
$< \quad (\texttt{13} + \text{bsearch1-t}\,(x,\ lst,\ 1 + ((i + j) \div \texttt{2}),\ j)))$
$= \quad \mathbf{f})$

THEOREM: bsearch1-t-ubound
$(\texttt{26} + (\texttt{13} * \log\,(\texttt{2},\ j - i))) \not< \text{bsearch1-t}\,(x,\ lst,\ i,\ j)$

EVENT: Disable bsearch1-t-crock.

THEOREM: bsearch-t-ubound-la
$\text{bsearch-t}\,(x,\ n,\ lst) \leq (\texttt{32} + (\texttt{13} * \log\,(\texttt{2},\ n)))$

THEOREM: bsearch-t-ubound
$(n < \exp\,(\texttt{2},\ \texttt{31})) \rightarrow (\text{bsearch-t}\,(x,\ n,\ lst) \leq \texttt{435})$

Index