```
#|

 Copyright (C) 1994 by Robert S. Boyer.  All Rights Reserved.

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

 NO WARRANTY

 Robert S. Boyer 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 Robert S. Boyer 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 initial **nqthm** theory.

EVENT: For efficiency, compile those definitions not yet compiled.

```
;   The floor of the square root. This definition is taken from Goodstein.
```

DEFINITION:
rt (x)
$= \ $ **if** $x \simeq 0$ **then** 0
elseif $((1 + \mathrm{rt}(x-1)) * (1 + \mathrm{rt}(x-1))) = x$ **then** $1 + \mathrm{rt}(x-1)$
else $\mathrm{rt}(x-1)$ **endif**

THEOREM: times-add1
$(x * (1 + y)) = (x + (x * y))$

THEOREM: plus-add1
$(x + (1 + y)) = (1 + (x + y))$

Theorem: square-0
$((x * x) = 0) = (x \simeq 0)$

Theorem: square-monotonic
$(b \not< a) \rightarrow ((b * b) \not< (a * a))$

Theorem: spec-for-rt
$(y \not< (\mathrm{rt}\,(y) * \mathrm{rt}\,(y)))$
$\wedge \quad (y < (1 + (\mathrm{rt}\,(y) + (\mathrm{rt}\,(y) + (\mathrm{rt}\,(y) * \mathrm{rt}\,(y))))))$

Theorem: rt-is-unique
$((a \in \mathbf{N}) \wedge ((a * a) \leq y) \wedge (y < ((1 + a) * (1 + a))))$
$\rightarrow \quad (a = \mathrm{rt}\,(y))$

Theorem: ncons-lemma
$\mathrm{rt}\,(u + ((u + v) * (u + v))) = (u + v)$

Definition: $\mathrm{ncar}\,(x) = (x - (\mathrm{rt}\,(x) * \mathrm{rt}\,(x)))$

Definition: $\mathrm{ncdr}\,(x) = (\mathrm{rt}\,(x) - \mathrm{ncar}\,(x))$

Definition: $\mathrm{ncons}\,(i, j) = (i + ((i + j) * (i + j)))$

Theorem: ncar-ncons
$(i \in \mathbf{N}) \rightarrow (\mathrm{ncar}\,(\mathrm{ncons}\,(i, j)) = i)$

Theorem: ncdr-ncons
$(j \in \mathbf{N}) \rightarrow (\mathrm{ncdr}\,(\mathrm{ncons}\,(i, j)) = j)$

Definition: $\mathrm{ncadr}\,(x) = \mathrm{ncar}\,(\mathrm{ncdr}\,(x))$

Definition: $\mathrm{ncaddr}\,(x) = \mathrm{ncar}\,(\mathrm{ncdr}\,(\mathrm{ncdr}\,(x)))$

Theorem: rt-lessp
$((x \not\simeq 0) \wedge (x \neq 1)) \rightarrow (\mathrm{rt}\,(x) < x)$

```
;   I'm sure the system could prove this without the hint, but it would use
;   induction and this is the obvious way to do it.
```

Theorem: rt-lesseqp
$x \not< \mathrm{rt}\,(x)$

Theorem: difference-0
$(x < y) \rightarrow ((x - y) = 0)$

THEOREM: lessp-difference-1
$((a - b) < c)$
$= \;\; \textbf{if } a < b \textbf{ then } 0 < c$
$\quad\quad \textbf{else } a < (b + c) \textbf{ endif}$

THEOREM: ncar-lesseqp
$x \not< \text{ncar}(x)$

THEOREM: crock
$(x < (\text{rt}(x) - d)) = \mathbf{f}$

THEOREM: ncdr-lesseqp
$x \not< \text{ncdr}(x)$

THEOREM: ncdr-lessp
$((fn \in \mathbf{N}) \land (\text{ncar}(fn) \neq 0) \land (\text{ncar}(fn) \neq 1)) \rightarrow (\text{ncdr}(fn) < fn)$

EVENT: Disable ncar.

EVENT: Disable ncdr.

DEFINITION:
pr-apply $(fn, \; arg)$

$$
\begin{array}{ll}
= & \textbf{if } fn \notin \mathbf{N} \textbf{ then } 0 \\
& \textbf{elseif } \text{ncar}(fn) = 0 \textbf{ then } 0 \\
& \textbf{elseif } \text{ncar}(fn) = 1 \textbf{ then } arg \\
& \textbf{elseif } \text{ncar}(fn) = 2 \textbf{ then } 1 + arg \\
& \textbf{elseif } \text{ncar}(fn) = 3 \textbf{ then } \text{rt}(arg) \\
& \textbf{elseif } \text{ncar}(fn) = 4 \\
& \textbf{then if } arg \simeq 0 \textbf{ then } 0 \\
& \qquad \textbf{else } \text{pr-apply}(\text{ncdr}(fn), \text{pr-apply}(fn, \; arg - 1)) \textbf{ endif} \\
& \textbf{elseif } \text{ncar}(fn) = 5 \\
& \textbf{then } \text{pr-apply}(\text{ncadr}(fn), \; arg) + \text{pr-apply}(\text{ncaddr}(fn), \; arg) \\
& \textbf{elseif } \text{ncar}(fn) = 6 \\
& \textbf{then } \text{pr-apply}(\text{ncadr}(fn), \; arg) - \text{pr-apply}(\text{ncaddr}(fn), \; arg) \\
& \textbf{elseif } \text{ncar}(fn) = 7 \\
& \textbf{then } \text{pr-apply}(\text{ncadr}(fn), \; arg) * \text{pr-apply}(\text{ncaddr}(fn), \; arg) \\
& \textbf{elseif } \text{ncar}(fn) = 8 \\
& \textbf{then } \text{pr-apply}(\text{ncadr}(fn), \text{pr-apply}(\text{ncaddr}(fn), \; arg)) \\
& \textbf{else } 0 \textbf{ endif}
\end{array}
$$

DEFINITION: non-pr-fn $(x) = (1 + \text{pr-apply}(x, \; x))$

DEFINITION: counter-example-for $(x) = \text{fix}(x)$

THEOREM: non-pr-functions-exist
non-pr-fn (counter-example-for (fn)) $\neq$ pr-apply (fn, counter-example-for (fn))

THEOREM: counter-example-for-numeric
counter-example-for (x) $\in \mathbf{N}$

EVENT: Disable pr-apply.

DEFINITION:
max2 (fn, i)
= **if** $i \simeq \mathtt{0}$ **then** pr-apply (fn, i)
else max (pr-apply (fn, i), max2 (fn, $i - 1$)) **endif**

DEFINITION:
max1 (fn, i)
= **if** $fn \simeq \mathtt{0}$ **then** max2 (fn, i)
else max (max2 (fn, i), max1 ($fn - 1$, i)) **endif**

THEOREM: max2-gte
max2 (i, j) $\not<$ pr-apply (i, j)

DEFINITION: exceed (j) = (1 + max1 (j, j))

DEFINITION: exceed-at (i) = i

THEOREM: max1-gte1
($fn \in \mathbf{N}$) $\rightarrow$ (max1 ($j + fn$, i) $\not<$ pr-apply (fn, i))

THEOREM: max1-gte2
($fn \in \mathbf{N}$) $\rightarrow$ (max1 ($j + fn$, $j + fn$) $\not<$ pr-apply (fn, $j + fn$))

THEOREM: exceed-is-bigger
($fn \in \mathbf{N}$)
$\rightarrow$ (pr-apply (fn, j + exceed-at (fn)) < exceed (j + exceed-at (fn)))

Index