```
#|
 Copyright (C) 1994 by Ken Kunen.  All Rights Reserved.

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

 NO WARRANTY

 Ken Kunen 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 Ken Kunen 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.

```
; load initial database

; This file was developed by Ken Kunen, kunen@cs.wisc.edu.  It checks
; the fundamental theorem of arithmetic.  Unlike the proof in the book
; A Compuational Logic and in the file basic.events, Kunen's proof does not
; need auxilliary functions, e.g., gcd.  Why should one be interested
; in this matter?

; In full first order logic, one can show that definitions are
; unnecessary, both in the ``use sense'' that they are ``eliminable''
; from theorems in which they occur and in the ``unnecessary'' sense
; that anything that can be proved with a definition (whose statement
; does not involve that definition) can be proved without the
; definition.

; However, NQTHM does not provide the full power of first order logic.
; It remains an open question, can any Nqthm theorem be proved without
; the definition of concepts not used in the theorem?  Kunen did the
```

```
; job for prime factorization.  Can someone else do the job for, say,
; Wilson's theorem?

; Foregoing comments by Robert S. Boyer

; file basic4 -- proof of prime-factorization uniqueness
; no definitions needed except those used to state the theorem,
; EXCEPT the identically 0 function kludge used to
; force the correct induction.
; events in caps were just copied from basic.events
; no attemt to be elegant here, but it works
; time on decstation3100:  [ 22.7 1448.2 29.1 ]



;;;;;;;;;;;  basic defns -- needed to state the uniqueness of prime fact.
```

DEFINITION: $\text{divides}(x,\ y) = ((y\ \textbf{mod}\ x) \simeq 0)$

DEFINITION:
$\text{prime1}(x,\ y)$
$=$ **if** $y \simeq 0$ **then f**
elseif $y = 1$ **then t**
else $(\neg\ \text{divides}(y,\ x)) \wedge \text{prime1}(x,\ y-1)$ **endif**

DEFINITION:
$\text{prime}(x) = ((x \not\simeq 0) \wedge (x \neq 1) \wedge \text{prime1}(x,\ x-1))$

DEFINITION:
$\text{delete}(x,\ y)$
$=$ **if** $\text{listp}(y)$
then if $x = \text{car}(y)$ **then** $\text{cdr}(y)$
else $\text{cons}(\text{car}(y),\ \text{delete}(x,\ \text{cdr}(y)))$ **endif**
else y **endif**

DEFINITION:
$\text{perm}(a,\ b)$
$=$ **if** $a \simeq$ **nil then** $b \simeq$ **nil**
elseif $\text{car}(a) \in b$ **then** $\text{perm}(\text{cdr}(a),\ \text{delete}(\text{car}(a),\ b))$
else f endif

DEFINITION:
$\text{prime-list}(l)$
$=$ **if** $l \simeq$ **nil then t**
else $\text{prime}(\text{car}(l)) \wedge \text{prime-list}(\text{cdr}(l))$ **endif**

DEFINITION:
times-list (l)
= **if** $l \simeq$ **nil** **then** 1
else car $(l) *$ times-list (cdr (l)) **endif**

```
; end defns
; THE GOAL
;    (PROVE-LEMMA PRIME-FACTORIZATION-UNIQUENESS NIL
;                 (IMPLIES (AND (PRIME-LIST L1)
;                               (PRIME-LIST L2)
;                               (EQUAL (TIMES-LIST L1)
;                                      (TIMES-LIST L2)))
;                          (PERM L1 L2)))
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;
;;;   Basic facts about +, *, -
;;;;;;;;;;;;;;;;;;;;;;;;
```

THEOREM: plus-right-id2
$(y \notin \mathbf{N}) \rightarrow ((x + y) = \text{fix}(x))$

THEOREM: plus-add1
$(x + (1 + y))$
= **if** $y \in \mathbf{N}$ **then** $1 + (x + y)$
else $1 + x$ **endif**

THEOREM: commutativity2-of-plus
$(x + (y + z)) = (y + (x + z))$

THEOREM: commutativity-of-plus
$(x + y) = (y + x)$

THEOREM: associativity-of-plus
$((x + y) + z) = (x + (y + z))$

THEOREM: plus-equal-0
$((a + b) = 0) = ((a \simeq 0) \wedge (b \simeq 0))$

THEOREM: difference-x-x
$(x - x) = 0$

THEOREM: difference-plus
$(((x + y) - x) = \text{fix}(y)) \wedge (((y + x) - x) = \text{fix}(y))$

THEOREM: plus-cancellation
$((a + b) = (a + c)) = (\text{fix}(b) = \text{fix}(c))$

THEOREM: difference-0
$(y \not< x) \rightarrow ((x - y) = 0)$

THEOREM: equal-difference-0
$(0 = (x - y)) = (y \not< x)$

THEOREM: difference-cancellation-0
$(x = (x - y)) = ((x \in \mathbf{N}) \wedge ((x = 0) \vee (y \simeq 0)))$

THEOREM: difference-cancellation-1
$((x - y) = (z - y))$
$=$ **if** $x < y$ **then** $y \not< z$
elseif $z < y$ **then** $y \not< x$
else $\text{fix}(x) = \text{fix}(z)$ **endif**

THEOREM: times-zero2
$(y \notin \mathbf{N}) \rightarrow ((x * y) = 0)$

THEOREM: distributivity-of-times-over-plus
$(x * (y + z)) = ((x * y) + (x * z))$

THEOREM: times-add1
$(x * (1 + y))$
$=$ **if** $y \in \mathbf{N}$ **then** $x + (x * y)$
else $\text{fix}(x)$ **endif**

THEOREM: commutativity-of-times
$(x * y) = (y * x)$

THEOREM: commutativity2-of-times
$(x * (y * z)) = (y * (x * z))$

THEOREM: associativity-of-times
$((x * y) * z) = (x * (y * z))$

THEOREM: equal-times-0
$((x * y) = 0) = ((x \simeq 0) \vee (y \simeq 0))$

THEOREM: equal-lessp
$((x < y) = z)$
$=$ **if** $x < y$ **then** $\mathbf{t} = z$
else $\mathbf{f} = z$ **endif**

THEOREM: difference-elim
$((y \in \mathbf{N}) \wedge (y \not< x)) \rightarrow ((x + (y - x)) = y)$

THEOREM: lessp-times-1
$(i \not\simeq \mathtt{0}) \rightarrow ((i * j) \not< j)$

THEOREM: lessp-times-2
$(i \not\simeq \mathtt{0}) \rightarrow ((j * i) \not< j)$

THEOREM: difference-plus1
$((x + y) - x) = \mathrm{fix}\,(y)$

THEOREM: difference-plus2
$((y + x) - x) = \mathrm{fix}\,(y)$

THEOREM: difference-plus-cancelation
$((x + y) - (x + z)) = (y - z)$

THEOREM: times-difference
$(x * (c - w)) = ((c * x) - (w * x))$

THEOREM: difference-plus3
$((b + (a + c)) - a) = (b + c)$

THEOREM: difference-add1-cancellation
$((\mathtt{1} + (y + z)) - z) = (\mathtt{1} + y)$

THEOREM: lessp-plus-cancelation
$((x + y) < (x + z)) = (y < z)$

THEOREM: lessp-times-cancellation
$((x * z) < (y * z)) = ((z \not\simeq \mathtt{0}) \wedge (x < y))$

THEOREM: lessp-plus-cancellation3
$(y < (x + y)) = (x \not\simeq \mathtt{0})$

THEOREM: times-id-iff-1
$(z = (w * z)) = ((z \in \mathbf{N}) \wedge ((z = \mathtt{0}) \vee (w = \mathtt{1})))$

THEOREM: times-identity1
$((y \in \mathbf{N}) \wedge (y \neq \mathtt{1}) \wedge (y \neq \mathtt{0}) \wedge (x \neq \mathtt{0})) \rightarrow (x \neq (x * y))$

THEOREM: times-identity
$(x = (x * y)) = ((x = \mathtt{0}) \vee ((x \in \mathbf{N}) \wedge (y = \mathtt{1})))$

THEOREM: times-equal-1
$((a * b) = 1)$
$= \quad ((a \neq 0)$
$\quad \wedge \quad (b \neq 0)$
$\quad \wedge \quad (a \in \mathbf{N})$
$\quad \wedge \quad (b \in \mathbf{N})$
$\quad \wedge \quad ((a - 1) = 0)$
$\quad \wedge \quad ((b - 1) = 0))$

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;  List facts -- pure lists only
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

THEOREM: delete-non-member
$(x \notin y) \rightarrow (\text{delete}\,(x,\, y) = y)$

THEOREM: member-delete
$(x \in \text{delete}\,(u,\, v)) \rightarrow (x \in v)$

THEOREM: commutativity-of-delete
$\text{delete}\,(x,\, \text{delete}\,(y,\, z)) = \text{delete}\,(y,\, \text{delete}\,(x,\, z))$

THEOREM: lessp-count-delete
$(n \in l) \rightarrow (\text{count}\,(\text{delete}\,(n,\, l)) < \text{count}\,(l))$

```
;;; ok to here -- save as basic4a
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;  Quotients, Remainders, Divisibility
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;   !!!!!!!  this one is really important
```

THEOREM: remainder-quotient
$((x \textbf{ mod } y) + (y * (x \div y))) = \text{fix}\,(x)$

THEOREM: remainder-wrt-1
$(y \textbf{ mod } 1) = 0$

THEOREM: remainder-wrt-12
$(x \notin \mathbf{N}) \rightarrow ((y \textbf{ mod } x) = \text{fix}\,(y))$

THEOREM: lessp-remainder2
$((x \textbf{ mod } y) < y) = (y \not\simeq 0)$

THEOREM: remainder-x-x
$(x \textbf{ mod } x) = 0$

THEOREM: remainder-quotient-elim
$((y \not\simeq 0) \wedge (x \in \mathbf{N})) \rightarrow (((x \textbf{ mod } y) + (y * (x \div y))) = x)$

THEOREM: lessp-quotient1
$((i \div j) < i) = ((i \not\simeq 0) \wedge ((j \simeq 0) \vee (j \neq 1)))$

THEOREM: lessp-remainder1
$((x \textbf{ mod } y) < x) = ((y \not\simeq 0) \wedge (x \not\simeq 0) \wedge (x \not< y))$

THEOREM: divides-times
$((x * z) \textbf{ mod } z) = 0$

THEOREM: remainder-add1
$((y \not\simeq 0) \wedge (y \neq 1)) \rightarrow (((1 + (x * y)) \textbf{ mod } y) \neq 0)$

THEOREM: divides-plus-rewrite1
$(((x \textbf{ mod } z) = 0) \wedge ((y \textbf{ mod } z) = 0)) \rightarrow (((x + y) \textbf{ mod } z) = 0)$

THEOREM: divides-plus-rewrite2
$(((x \textbf{ mod } z) = 0) \wedge ((y \textbf{ mod } z) \neq 0)) \rightarrow (((x + y) \textbf{ mod } z) \neq 0)$

THEOREM: divides-plus-rewrite
$((x \textbf{ mod } z) = 0) \rightarrow ((((x + y) \textbf{ mod } z) = 0) = ((y \textbf{ mod } z) = 0))$

THEOREM: divides-plus-rewrite-commuted
$((x \textbf{ mod } z) = 0) \rightarrow ((((y + x) \textbf{ mod } z) = 0) = ((y \textbf{ mod } z) = 0))$

THEOREM: euclid
$((x \textbf{ mod } z) = 0)$
$\rightarrow \quad ((((y - x) \textbf{ mod } z) = 0)$
$\quad = \quad \textbf{if } x < y \textbf{ then } (y \textbf{ mod } z) = 0$
$\quad\quad \textbf{else t endif})$

THEOREM: remainder-0-crock
$(0 \textbf{ mod } y) = 0$

THEOREM: quotient-times1
$((y \in \mathbf{N}) \wedge (x \in \mathbf{N}) \wedge (x \neq 0) \wedge \text{divides}\,(x,\, y))$
$\rightarrow \quad ((x * (y \div x)) = y)$

THEOREM: quotient-lessp
$((x \not\simeq 0) \wedge (x < y)) \rightarrow ((y \div x) \neq 0)$

THEOREM: divides-times1
$(a = (z * y)) \rightarrow ((a \textbf{ mod } z) = 0)$

THEOREM: quotient-divides
$((y \in \mathbf{N}) \wedge ((x * (y \div x)) \neq y)) \rightarrow ((y \textbf{ mod } x) \neq 0)$

THEOREM: quotient-times
$((y * x) \div y)$
$=$ **if** $y \simeq 0$ **then** 0
else $\mathrm{fix}(x)$ **endif**

THEOREM: distributivity-of-divides
$((a \not\simeq 0) \wedge \mathrm{divides}(a, w)) \rightarrow ((c * (w \div a)) = ((c * w) \div a))$

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;  Times-list and Prime-list facts -- only stuff that
;;;  doesn't depend on meaning of "prime"
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

THEOREM: times-list-append
$\mathrm{times\text{-}list}(\mathrm{append}(x, y)) = (\mathrm{times\text{-}list}(x) * \mathrm{times\text{-}list}(y))$

THEOREM: prime-list-append
$\mathrm{prime\text{-}list}(\mathrm{append}(x, y)) = (\mathrm{prime\text{-}list}(x) \wedge \mathrm{prime\text{-}list}(y))$

THEOREM: prime-list-delete
$\mathrm{prime\text{-}list}(l2) \rightarrow \mathrm{prime\text{-}list}(\mathrm{delete}(x, l2))$

```
;  ok to here -- save as "basic4b"


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;  Primes and their properties
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; as far as possible, try to express theorems in terms of
; divisibility, rather than remainder=0 -- then, in later lemmas,
; we don't have to expand DIVIDES
; ????????? actually, that's not so clear -- divides is still
; expanded, even when it's disabled
```

THEOREM: primes-are-big
$\text{prime}(p) \rightarrow ((1 < p) = \mathbf{t})$

THEOREM: little-step-aux1
$(\text{prime1}(p, z) \wedge (1 < y) \wedge (y \leq z)) \rightarrow (\neg\ \text{divides}(y, p))$

THEOREM: little-step-aux2
$(\text{prime}(p) \wedge (1 < y) \wedge (y < p)) \rightarrow (\neg\ \text{divides}(y, p))$

THEOREM: little-step-aux3
$((1 < p) \wedge (1 < y) \wedge \text{divides}(y, p)) \rightarrow (y \leq p)$

THEOREM: little-step
$(\text{prime}(p) \wedge (1 < y) \wedge (y \neq p)) \rightarrow (\neg\ \text{divides}(y, p))$

THEOREM: exact-remainder-quotient
$((x \in \mathbf{N}) \wedge \text{divides}(p, x)) \rightarrow ((p * (x \div p)) = x)$

THEOREM: divides-product-aux1
$\text{divides}(p, x) \rightarrow ((x * y) = (p * ((x \div p) * y)))$

THEOREM: divides-product-aux2
$\text{divides}(p, p * w) = \mathbf{t}$

THEOREM: divides-product
$\text{divides}(p, x) \rightarrow \text{divides}(p, x * y)$

```
; ok to here -- saved as basic4c
```

THEOREM: divides-sum
$(\text{divides}(p, u + v) \wedge \text{divides}(p, v)) \rightarrow \text{divides}(p, u)$

```
; really just a restatement of DIVIDES-PLUS-REWRITE
```

THEOREM: divides-reduction-aux1
$(1 < p) \rightarrow (((p\ \mathbf{mod}\ b) + (b * (p \div b))) = p)$

```
; just a restatement of REMAINDER-QUOTIENT
```

```
; now, multiply on left by a:
```

THEOREM: divides-reduction-aux2
$(1 < p)$
$\rightarrow \quad (((a * (p\ \mathbf{mod}\ b)) + (a * (b * (p \div b)))) = (a * p))$

; since p divides a*p:

THEOREM: divides-reduction-aux3
$(1 < p) \rightarrow \text{divides}\,(p,\, (a * (p \bmod b)) + (a * (b * (p \div b))))$

; now, if p divides a*b, it divides the second term

THEOREM: divides-reduction-aux4
$\text{divides}\,(p,\, a * b) \rightarrow \text{divides}\,(p,\, a * (b * \mathit{anything}))$

; hence:

THEOREM: divides-reduction
$((1 < p) \wedge \text{divides}\,(p,\, a * b)) \rightarrow \text{divides}\,(p,\, a * (p \bmod b))$

; point of this lemma -- it reduces b to a smaller b', so induction can be used

; basis for induction:

THEOREM: prime-divides-small-product-aux1
$\text{divides}\,(p,\, a * 1) \rightarrow \text{divides}\,(p,\, a)$

; ok

THEOREM: prime-divides-small-product-aux2
$(\text{prime}\,(p) \wedge (1 < b) \wedge (b < p)) \rightarrow (0 < (p \bmod b))$

; just a restatement of little-step

; ok to here -- save as "basic4d"

; goal -- prime-divides-small-product is of form:
; (prime p) --> phi(b,p,a)
; we want to prove a lemma saying (prime p) & not(phi(b,p,a)) -->
; remainder(p,b) < b and not(phi(remainder(p,b),p,a))
; phi is: not(b<p and 0<b and divides(p, a*b) and not(divides(p, a)))
; we can't seem to prove the induction lemma all in one step

THEOREM: prime-divides-small-product-aux3
$(0 < b) \rightarrow ((1 < b) \vee (1 = b))$

; ok

THEOREM: prime-divides-small-product-aux4
$((0 < b) \wedge \text{divides}(p,\ a * b) \wedge (\neg\ \text{divides}(p,\ a))) \rightarrow (1 < b)$

```
; ok
; i.e., the the induction lemma is trivial unless 1<b
```

THEOREM: prime-divides-small-product-aux5
$(\text{prime}(p) \wedge (1 < b)) \rightarrow ((p\ \textbf{mod}\ b) < b)$

```
; ok
```

THEOREM: prime-divides-small-product-aux6
$(\text{prime}(p) \wedge (1 < b) \wedge (b < p)) \rightarrow (0 < (p\ \textbf{mod}\ b))$

```
; ok
```

THEOREM: prime-divides-small-product-aux7
$(\text{prime}(p) \wedge (1 < b) \wedge (b < p)) \rightarrow ((p\ \textbf{mod}\ b) < p)$

```
; ok
```

THEOREM: prime-divides-small-product-aux8
$$
\begin{array}{ll}
(\text{prime}(p) & \\
\wedge & (1 < b) \\
\wedge & (b < p) \\
\wedge & \text{divides}(p,\ a * b) \\
\wedge & (\neg\ \text{divides}(p,\ a))) \\
\rightarrow & (((p\ \textbf{mod}\ b) < b) \\
& \quad \wedge \quad (0 < (p\ \textbf{mod}\ b)) \\
& \quad \wedge \quad ((p\ \textbf{mod}\ b) < p) \\
& \quad \wedge \quad \text{divides}(p,\ a * (p\ \textbf{mod}\ b)) \\
& \quad \wedge \quad (\neg\ \text{divides}(p,\ a)))
\end{array}
$$

```
; trivially rewrites to T, after 166.6 seconds !!!!!!!!

; the induction lemma -- same as above, but with 1 replaced by 0
```

THEOREM: prime-divides-small-product-aux9
$$
\begin{array}{ll}
(\text{prime}(p) & \\
\wedge & (0 < b) \\
\wedge & (b < p) \\
\wedge & \text{divides}(p,\ a * b)
\end{array}
$$

$\wedge$ $(\neg\ \text{divides}\,(p,\ a)))$
$\rightarrow$ $(((p\ \textbf{mod}\ b) < b)$
$\wedge$ $(0 < (p\ \textbf{mod}\ b))$
$\wedge$ $((p\ \textbf{mod}\ b) < p)$
$\wedge$ $\text{divides}\,(p,\ a * (p\ \textbf{mod}\ b))$
$\wedge$ $(\neg\ \text{divides}\,(p,\ a)))$

```
; rewrites trivially to T after 76.3 sec


; we still have to force nqthm to do the right induction
```

DEFINITION:
kludge $(b,\ p)$
$=$ **if** $\neg$ prime (p) **then** 0
elseif $0 \not< b$ **then** 0
else kludge $(p\ \textbf{mod}\ b,\ p)$ **endif**

```
; always returns 0, but now we can use it in an induct hint
```

THEOREM: prime-divides-small-product
$(\text{prime}\,(p) \wedge \text{divides}\,(p,\ a * b) \wedge (0 < b) \wedge (b < p))$
$\rightarrow$ $\text{divides}\,(p,\ a)$

```
; ok after 199.1 sec -- probably phrasing some of the auxilliary
; lemmas as rewrite rules would have helped speed things up

;;; ok to here -- save as basic4e

; now, try to replace (b < p) by (not (divides p b))
; idea -- if p divdes a*b, then p divides a*(b mod p) -- if  b mod p
; isn't 0, then it's between 0 and p, so apply prime-divides-small-product

; first step -- prove a variant of divides-reduction, but replacing
; (remainder p b) by (remainder b p)
; i.e.  : divides p (a * b) --> divides p (a * remainder b p)
; say,
; plan
; remainder b p + p * b/p  = fix( b)    (REMAINDER-QUOTENT)
;  a * remainder b p + a * ( p * b/p) = a * b
```

THEOREM: divides-reduction-var-aux1
$(b \in \mathbf{N})$
$\rightarrow \quad (((a * (b \textbf{ mod } p)) + (a * (p * (b \div p)))) = (a * b))$

THEOREM: divides-reduction-var-aux2
$\text{divides}\,(p,\ p * \textit{anything})$

THEOREM: divides-reduction-var-aux3
$(a * (p * \textit{anything})) = (p * (a * \textit{anything}))$

THEOREM: divides-reduction-var-aux4
$\text{divides}\,(p,\ a * (p * \textit{anything}))$

THEOREM: divides-reduction-var
$((b \in \mathbf{N}) \wedge \text{divides}\,(p,\ a * b)) \rightarrow \text{divides}\,(p,\ a * (b \textbf{ mod } p))$

```
; ok to here -- save as basic4f
```

THEOREM: prime-divides-product-aux1
$((b \textbf{ mod } p) \neq \texttt{0}) \rightarrow ((\texttt{0} < (b \textbf{ mod } p)) = \mathbf{t})$

THEOREM: prime-divides-product-aux2
$\text{prime}\,(p) \rightarrow (((b \textbf{ mod } p) < p) = \mathbf{t})$

THEOREM: prime-divides-product
$(\text{prime}\,(p) \wedge \text{divides}\,(p,\ a * b) \wedge (\neg\ \text{divides}\,(p,\ b))) \rightarrow \text{divides}\,(p,\ a)$

```
;;;;;;;;  THIS is the key lemma -- the rest should be just list-hacking
```

THEOREM: car-divides-times-list
$\text{listp}\,(l) \rightarrow \text{divides}\,(\text{car}\,(l),\ \text{times-list}\,(l))$

THEOREM: divides-equal
$(\text{prime}\,(p) \wedge \text{prime}\,(q) \wedge \text{divides}\,(p,\ q)) \rightarrow ((p = q) = \mathbf{t})$

```
; following should be trivial by times-divides-product
```

THEOREM: prime-divides-list-aux1
$(\text{prime-list}\,(l)$
$\wedge \quad \text{prime}\,(p)$
$\wedge \quad \text{listp}\,(l)$
$\wedge \quad \text{divides}\,(p,\ \text{times-list}\,(l))$
$\wedge \quad (\neg\ \text{divides}\,(p,\ \text{times-list}\,(\text{cdr}\,(l)))))$
$\rightarrow \quad \text{divides}\,(p,\ \text{car}\,(l))$

EVENT: Disable prime-divides-small-product-aux1.

EVENT: Disable prime-divides-small-product-aux2.

EVENT: Disable prime-divides-small-product.

```
; these auxilliaries are getting in the way
```

THEOREM: prime-divides-list-aux2
$(\text{prime-list}\,(l)$
$\wedge \quad \text{prime}\,(p)$
$\wedge \quad \text{listp}\,(l)$
$\wedge \quad \text{divides}\,(p, \text{times-list}\,(l))$
$\wedge \quad (\neg\ \text{divides}\,(p, \text{times-list}\,(\text{cdr}\,(l)))))$
$\rightarrow \quad (\text{car}\,(l) = p)$

```
; ok
```

EVENT: Disable prime-divides-list-aux1.

```
; basis of induction
```

THEOREM: prime-divides-list-aux3
$(\text{prime}\,(p) \wedge (l \simeq \textbf{nil})) \rightarrow (\neg\ \text{divides}\,(p, \text{times-list}\,(l)))$

```
; ok
```

```
; so it can see the trivial induction:
```

THEOREM: prime-divides-list-aux4
$(\text{prime-list}\,(l) \wedge \text{prime}\,(p) \wedge \text{divides}\,(p, \text{times-list}\,(l)))$
$\rightarrow \quad (\text{listp}\,(l) \wedge ((\text{car}\,(l) = p) \vee \text{divides}\,(p, \text{times-list}\,(\text{cdr}\,(l)))))$

```
; ok
```

EVENT: Disable prime-divides-list-aux2.

EVENT: Disable prime-divides-list-aux3.

```
; now, by a simple induction on L:
```

THEOREM: prime-divides-list
$(\text{prime-list}\,(l) \wedge \text{prime}\,(p) \wedge \text{divides}\,(p, \text{times-list}\,(l))) \rightarrow (p \in l)$

```
; ok
```

```
;;; two lemmas from basic.events now follow trivially
```

THEOREM: prime-list-times-list
$(\text{prime}\,(c) \wedge \text{prime-list}\,(l2) \wedge (c \notin l2))$
$\rightarrow \quad ((\text{times-list}\,(l2)\ \textbf{mod}\ c) \neq 0)$

THEOREM: prime-member
$(((c * \text{times-list}\,(l1)) = \text{times-list}\,(l2)) \wedge \text{prime}\,(c) \wedge \text{prime-list}\,(l2))$
$\rightarrow \quad (c \in l2)$

```
;;;;; some more lemmas about lists and producs
```

```
; this is the base case in PRIME-FACTORIZATION-UNIQUENESS
```

THEOREM: void-case
$(\text{prime-list}\,(l) \wedge (1 = \text{times-list}\,(l))) \rightarrow (l \simeq \textbf{nil})$

```
; ok
```

THEOREM: product-delete
$(x \in l) \rightarrow (\text{times-list}\,(l) = (x * \text{times-list}\,(\text{delete}\,(x, l))))$

```
; ok
```

THEOREM: times-cancellation
$((0 < p) \wedge ((p * x) = (p * y))) \rightarrow (\text{fix}\,(x) = \text{fix}\,(y))$

```
; ok
```

THEOREM: number-times-cancellation
$((x \in \mathbf{N}) \wedge (y \in \mathbf{N}) \wedge (0 < p) \wedge ((p * x) = (p * y)))$
$\rightarrow \quad ((x = y) = \mathbf{t})$

```
; ok
```

THEOREM: products-are-numbers
$(\text{times-list}\,(l) \in \mathbf{N}) = \mathbf{t}$

```
; ok

; changed
```

THEOREM: divide-by-member-aux1
$((p \in l) \wedge ((p * x) = \text{times-list}\,(l)))$
$\rightarrow \quad ((p * \text{times-list}\,(\text{delete}\,(p,\, l))) = (p * x))$

```
; ok

; next is just a kludge to force final rewrite
; with hypotheses stated in exact form required
```

THEOREM: divide-by-member-aux2
$(((p * x) = \text{times-list}\,(l))$
$\wedge \quad (x \in \mathbf{N})$
$\wedge \quad (p \neq \mathtt{0})$
$\wedge \quad (p \in \mathbf{N})$
$\wedge \quad (\text{times-list}\,(l) = (p * \text{times-list}\,(\text{delete}\,(p,\, l)))))$
$\rightarrow \quad (\text{times-list}\,(\text{delete}\,(p,\, l)) = x)$

```
; trivially, we should have now:
```

THEOREM: divide-by-member
$((x \in \mathbf{N}) \wedge (p \in l) \wedge (\mathtt{0} < p) \wedge ((p * x) = \text{times-list}\,(l)))$
$\rightarrow \quad (\text{times-list}\,(\text{delete}\,(p,\, l)) = x)$

```
; ok
```

THEOREM: divide-by-prime-member-aux1
$(\text{prime-list}\,(l) \wedge (p \in l)) \rightarrow \text{prime}\,(p)$

```
; since primes are positive:
```

THEOREM: divide-by-prime-member
$((x \in \mathbf{N}) \wedge \text{prime-list}\,(l) \wedge (p \in l) \wedge ((p * x) = \text{times-list}\,(l)))$
$\rightarrow \quad (\text{times-list}\,(\text{delete}\,(p,\, l)) = x)$

```
; ok

; should handle induction case in prime factorization uniqueness
```

THEOREM: reduct-product
(prime-list (*l1*)
 ∧ prime-list (*l2*)
 ∧ (times-list (*l1*) = times-list (*l2*))
 ∧ listp (*l1*)
 ∧ (car (*l1*) ∈ *l2*))
→ (times-list (delete (car (*l1*), *l2*)) = times-list (cdr (*l1*)))

```
; ok
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; On to the goal:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

EVENT: Disable divides-equal.

EVENT: Disable divide-by-prime-member-aux1.

THEOREM: prime-factorization-uniqueness-aux1
(prime-list (*l*) ∧ listp (*l*)) → (1 < times-list (*l*))

```
; a trivial variant:
```

THEOREM: prime-factorization-uniqueness-aux2
(prime-list (*l*) ∧ (1 = times-list (*l*))) → (¬ listp (*l*))

EVENT: Disable product-delete.

```
; seems to require some hints in our present setting
```

THEOREM: prime-factorization-uniqueness
(prime-list (*l1*) ∧ prime-list (*l2*) ∧ (times-list (*l1*) = times-list (*l2*)))
→ perm (*l1*, *l2*)

Index