1. Prove that whenever \( p_1, \ldots, p_n \) is a list of two or more propositions,

\[ \neg(p_1 \lor p_2 \lor \cdots \lor p_n) \]

is logically equivalent to

\[ \neg p_1 \land \neg p_2 \land \cdots \land \neg p_n \]

Use mathematical induction, and the fact that \( \neg(p \land q) \) is equivalent to \( \neg p \lor \neg q \) (De Morgan’s law).

**Answer:**

Proof. Basis step: \( n = 2 \). In this case, \( \neg(p_1 \lor p_2) \) is equivalent to \( \neg p_1 \land \neg p_2 \) by De Morgan.

Inductive step. Suppose \( \neg(p_1 \lor p_2 \lor \cdots \lor p_k) \) is equivalent to \( \neg p_1 \land \neg p_2 \land \cdots \land \neg p_k \). Consider \( \neg(p_1 \lor p_2 \lor \cdots \lor p_k \lor p_{k+1}) \). By De Morgan, this is equivalent to \( \neg(p_1 \lor p_2 \lor \cdots \lor p_k) \land \neg p_{k+1} \). By the induction hypothesis, this is equivalent to \( \neg p_1 \land \neg p_2 \land \cdots \land \neg p_k \land \neg p_{k+1} \). This is what we had to show.

We complete the proof by mathematical induction.
2. Prove by induction that if \( a \equiv b \pmod{m} \) then \( a^n \equiv b^n \pmod{m} \) for all \( n \geq 0 \).

**Answer:**
Proof. The key fact for the proof is that if \( a \equiv b \pmod{m} \) and \( c \equiv d \pmod{m} \) then \( ac \equiv bd \pmod{m} \). This is Theorem 10 in the text on page 163.

Basis step: \( n = 0 \). In this case, \( a^0 = 1 \) and \( b^0 = 1 \) and \( 1 \equiv 1 \pmod{m} \).

Inductive step. Suppose \( a^k \equiv b^k \pmod{m} \). Consider \( a^{k+1} = a(a^k) \) and \( b^{k+1} = b(b^k) \). Since \( a \equiv b \pmod{m} \) and by hypothesis \( a^k \equiv b^k \pmod{m} \), by Theorem 10, \( a^{k+1} \equiv b^{k+1} \pmod{m} \).

We complete the proof by mathematical induction.
3. Verify that the program segment

```plaintext
if x < y then
  m := x
else
  m := y
```

is correct with respect to the initial assertion \( T \) and the final assertion

\[
(x \leq y \land m = x) \lor (x > y \land m = y)
\]

**Answer:**
We use the rule for verifying conditional programs by verifying each branch. We consider the branches in turn.

First, it must be shown that if the initial assertion is true and \( x < y \), then after we execute \( m := x \), it’s true that \((x \leq y \land m = x) \lor (x > y \land m = y)\). By inertia, \( x < y \) is true after we execute \( m := x \). And by implication, \( x \leq y \) is true. By assignment, \( m = x \) is true after we execute \( m := x \). So by logic, \( x \leq y \land m = x \) is true and thus \((x \leq y \land m = x) \lor (x > y \land m = y)\).

Second, it must be shown that if the initial assertion is true and \( y \leq x \), then after we execute \( m := y \), it’s true that \((x \leq y \land m = x) \lor (x > y \land m = y)\). By inertia, \( y \leq x \) is true after we execute \( m := y \). By assignment, \( m = y \) is true after we execute \( m := y \). So we have \( y \leq x \land m = y \). We consider cases for \( y \leq x \): either \( x > y \) or \( y = x \). So either \((x > y \land m = y) \lor (y = x \land m = y)\). Looking at the second disjunct, \( y = x \land m = y \) is equivalent to \( y = x \land m = x \) and thus entails \( x \leq y \land m = x \). So we conclude \((x \leq y \land m = x) \lor (x > y \land m = y)\).

This completes the proof.
4. This program computes quotients and remainders:

\[
\begin{align*}
  r &:= a \\
  q &:= 0 \\
  \textbf{while } r \geq d \\
  \textbf{begin} \\
  & r := r - d \\
  & q := q + 1 \\
  \textbf{end}
\end{align*}
\]

The program assumes that \( d > 0 \) and \( a > 0 \).

Prove that

\[
d > 0 \land 0 \leq r \leq a \land a = dq + r
\]

is a loop invariant for the \texttt{while} loop. In other words, show that if

\[
d > 0 \land 0 \leq r \leq a \land a = dq + r \land r \geq d
\]

is true at the beginning of any iteration of the loop, then

\[
d > 0 \land 0 \leq r \leq a \land a = dq + r
\]

is true afterwards.

\textbf{Answer:}

Each iteration of the loop carries out the two instructions \( I_1 \) of \( r := r - d \) and \( I_2 \) of \( q := q + 1 \). First we show that if \( d > 0 \land 0 \leq r \leq a \land a = dq + r \) is true before \( I_1 \) then \( d > 0 \land 0 \leq r \leq a \land a = dq + r \) is true afterwards. Since \( d \) does not change in \( I_1 \), and we know \( d > 0 \) before \( I_1 \), we know \( d > 0 \) after \( I_1 \). Initially \( r \) has some value: call it \( r_0 \). We know initially \( a = dq + r_0 \). This does not depend on \( r \), so it holds after \( I_1 \) by inertia. Likewise \( 0 \leq r_0 \leq a \) holds after \( I_1 \) by inertia. Meanwhile, by assignment, we know that after \( I_1 \), \( r = r_0 - d \). Since \( d > 0 \) and \( d \leq r_0 < a \) we know \( 0 \leq r < a \) after \( I_1 \). Finally, by algebra \( r_0 = r + d \) and thus \( a = dq + r + d \). By logic then, after \( I_1 \), \( d > 0 \land 0 \leq r \leq a \land a = dq + r + d \).

Next we show that if \( d > 0 \land 0 \leq r \leq a \land a = dq + r + d \) is true before \( I_2 \), then \( d > 0 \land 0 \leq r \leq a \land a = dq + r \) is true afterwards. Neither \( d \) nor \( r \) nor \( a \) is affected by assignment to \( q \) so that means \( d > 0 \land 0 \leq r \leq a \) is true after \( I_2 \). Before \( I_2 \) \( q \) has some value: call it \( q_0 \). We know initially \( a = dq_0 + r + d \). This does not depend on \( q \), so it holds after \( I_2 \) by inertia. Meanwhile, by assignment, we know that after \( I_2 \), \( q = q_0 + 1 \). By algebra \( q_0 = q - 1 \) so \( a = dq_0 + r + d = dq - d + r + d = dq + r \). Thus by logic \( d > 0 \land 0 \leq r \leq a \land a = dq + r \) is true after \( I_2 \).

We complete the proof by observing that since the two instructions are run in sequence, these two arguments suffice to show that if \( d > 0 \land 0 \leq r \leq a \land a = dq + r \land r \geq d \) is true before any iteration, then \( d > 0 \land 0 \leq r \leq a \land a = dq + r \) is true afterwards.
5. Briefly, why does this invariant guarantee that the program can only terminate with a correct answer.

**Answer:**
When the loop completes, we have $a = dq + r$ and $0 \leq r$ by the loop invariant and $r < d$ by the termination condition. That makes $r$ the remainder and $q$ the quotient, by the Division Algorithm.