Problem 1: Let $(\mathbb{N},+)$ be the structure with domain the set $\mathbb{N}$ of all non-negative integers with the binary functions $+,$ respectively.

(a) Write $(\forall x)(\forall y)(\forall z) (x + y = y + z \implies x = z)$ in English as an English sentence.

(b) Write the proposition “Every non-negative integer can be represented as the sum of four squares” in predicate logic.

Solution:

(a) “For any three non-negative integers, if the sum of the first two equals the sum of the last two, then the first must equal the third.”

(b) $(\forall x)(\exists y)(\exists z)(\exists u)(\exists v) (x = y \cdot y + z \cdot z + u \cdot u + v \cdot v)$.

Problem 2: Construct a predicate tableau proof of

(a) $(\forall x)((\varphi(x) \land \psi(x)) \rightarrow (\forall x)(\varphi(x)) \land (\forall x)(\psi(x)))$

(b) $(\neg(\forall x)(\exists y)\varphi(x,y)) \rightarrow (\exists x)(\forall y)(\neg \varphi(x,y))$

Solution:

(a) \begin{align*}
&F (\forall x)(\varphi(x) \land \psi(x)) \rightarrow (\forall x)(\varphi(x)) \land (\forall x)(\psi(x)) \\
&\quad | T (\forall x)(\varphi(x) \land \psi(x)) \\
&\quad | F (\forall x)(\varphi(x)) \land (\forall x)(\psi(x)) \\
&\quad | F (\forall x)(\varphi(x)) \quad F (\forall x)(\psi(x)) \\
&\quad | F \varphi(c) \quad F \psi(d) \\
&\quad | T (\varphi(c) \land \psi(c)) \\
&\quad | T \varphi(c) \\
&\quad | T \psi(c) \\
&\quad \otimes \\
&\quad \otimes
\end{align*}
(b) \[ F \left( \neg(\forall x)(\exists y)\varphi(x, y) \right) \rightarrow (\exists x)(\forall y)(\neg\varphi(x, y)) \]
\[
\begin{align*}
T \ (\neg(\forall x)(\exists y)\varphi(x, y)) \\
F \ (\exists x)(\forall y)(\neg\varphi(x, y)) \\
F \ (\forall x)(\exists y)\varphi(x, y) \\
F \ (\exists y)\varphi(c, y) \\
F \ (\forall y)(\neg\varphi(c, y)) \\
F \ (\neg\varphi(c, d)) \\
F \ (\varphi(c, d)) \\
T \ (\varphi(c, d))
\end{align*}
\]

\[ \otimes \]

**Problem 3:** Convert into clausal form: \( (p \land \neg q) \lor (q \land \neg p) \).

**Solution:** The disjunction distributes over the conjunctions to give the equivalent
\[
(p \lor q) \land (p \lor \neg p) \land (\neg q \lor q) \land (\neg q \lor \neg p)
\]
And in clausal form:
\[
\{ \{p, q\}, \{p, \neg p\}, \{\neg q, q\}, \{\neg q, \neg p\} \}
\]

**Problem 4:** Give a resolution proof: \( (p \land (q \lor r)) \rightarrow ((p \land q) \lor (p \land r)) \).

**Solution:**
\[
\neg ((p \land (q \lor r)) \rightarrow ((p \land q) \lor (p \land r)))
\]
\[
\iff (p \land (q \lor r)) \land \neg ((p \land q) \lor (p \land r))
\]
\[
\iff p \land (q \lor r) \land (\neg p \lor \neg q) \land (\neg p \lor \neg r)
\]
And in clausal form:
\[
\{ \{p\}, \{q, r\}, \{\neg p, \neg q\}, \{\neg p, \neg r\} \}
\]
We produce a resolution refutation:
\[
\{q, r\} \rightarrow \{\neg p, \neg r\} \rightarrow \{p\} \rightarrow \{\neg q\} \rightarrow \{r, \neg p\} \rightarrow \{\neg p\} \rightarrow \{\neg p, \neg r\} \rightarrow \square
\]
And thus the original formula is shown to be valid.
Problem 5: Can the following terms be unified? If there is a most general unifier, find it, otherwise explain why the terms cannot be unified.

(a) \( f(a, g(x)) \) and \( f(y, g(a)) \)

(b) \( f(g(h(x)), x) \) and \( f(h(x), x) \)

Solution:

(a) The substitution \( \{ y/a, x/a \} \) is a most general unifier for the set \( \{ f(a, g(x)), f(y, g(a)) \} \). Applying it to both expressions produces the expression \( f(a, g(a)) \).

(b) The set \( \{ f(g(h(x)), x), f(h(x), x) \} \) has no unifier, since the first arguments \( (g(h(x)), h(x)) \) begin with different function names.

Problem 6: Give a resolution proof:

\[
-(\forall x)(\exists y)\varphi(x, y) \rightarrow (\exists x)(\forall y)(\neg \varphi(x, y)).
\]

Solution:

\[
\neg ((\forall x)(\exists y)\varphi(x, y)) \rightarrow (\exists x)(\forall y)(\neg \varphi(x, y))
\]

\[
\iff (\forall x)(\exists y)\varphi(x, y) \land (\exists x)(\forall y)(\neg \varphi(x, y))
\]

\[
\iff (\exists x)(\forall y)(\neg \varphi(x, y)) \land (\forall x)(\exists y)\varphi(x, y)
\]

We Skolemize this (introducing a constant symbol \( c \) and a unary function symbol \( f \)) and get \( (\neg \varphi(c, y)) \land \varphi(x, f(x)) \), or in clausal form

\[
\{ \neg \varphi(c, y) \}, \{ \varphi(x, f(x)) \}
\]

To perform a resolution, we first need to unify the set \( \{ \varphi(c, y), \varphi(x, f(x)) \} \). This can be done with the substitution \( \{ x/c, y/f(c) \} \). Applying it, the formula becomes

\[
\{ \neg \varphi(c, f(c)) \}, \{ \varphi(c, f(c)) \}
\]

and trivially resolves to give the empty clause.

Problem 7: You have a firm of translators. Each translator in the firm speaks two different languages, according to the following Prolog knowledge base:

\[
\text{translator_speaks(english, german).}
\]
\[
\text{translator_speaks(english, french).}
\]
\[
\text{translator_speaks(english, spanish).}
\]
\[
\text{translator_speaks(japanese, hebrew).}
\]
\[
\text{translator_speaks(spanish, italian).}
\]
\[
\text{translator_speaks(spanish, portuguese).}
\]

(a) Define a binary predicate \texttt{can_translate_quickly} which will succeed exactly when given two different languages which are known by a single translator in your firm.
(b) We would like to define a binary predicate can_translate which will be true exactly when given two languages that your firm can translate between. Here is a first attempt:

\[
\text{can\_translate}(X,Y) \leftarrow \text{can\_translate\_quickly}(X,Y).
\]
\[
\text{can\_translate}(X,Y) \leftarrow \text{can\_translate\_quickly}(X,Z), \text{can\_translate}(Z,Y).
\]

What is wrong with this definition?

(c) Here is a better attempt:

\[
\text{tnt}(X,X,1).
\]
\[
\text{tnt}(X,Y,1) \leftarrow \text{can\_translate\_quickly}(X,Y).
\]
\[
\text{tnt}(X,Y,N) \leftarrow N>1, M \text{ is } N-1, \text{tnt}(X,Z,1), \text{tnt}(Z,Y,M).
\]
\[
\text{can\_translate}(X,Y) \leftarrow \text{tnt}(X,Y,6).
\]

Explain what the ternary predicate tnt means, and why this implementation works correctly. Why does the number 6 appear where it does in the last line? Can it be replaced with bigger numbers? With smaller numbers?

Solution:

(a) We would like can\_translate\_quickly to accept any pair of languages which translator\_speaks, in either order. The following does the trick:

\[
\text{can\_translate\_quickly}(X,Y) \leftarrow \text{translator\_speaks}(X,Y).
\]
\[
\text{can\_translate\_quickly}(X,Y) \leftarrow \text{translator\_speaks}(Y,X).
\]

(b) This definition has declarative correctness, but not a procedural one. Generally speaking, the problem is that the graph of can\_translate\_quickly contains cycles (because of its symmetry). These cycles may cause the recursive definition above to loop forever in search of a path. Specifically, when given the query

?- can\_translate(german, portuguese).

prolog will do the following:

1. Try to satisfy can\_translate\_quickly(german, portuguese). This fails, of course.
2. Try to satisfy can\_translate\_quickly(german, Z), and then can\_translate(Z, portuguese).
3. The first of the two will be satisfied by assigning Z the value english, and now try to satisfy can\_translate(english, portuguese).
4. Try to satisfy can\_translate\_quickly(english, portuguese). This fails, of course.
5. Try to satisfy can\_translate\_quickly(english, Z), and then can\_translate(Z, portuguese).
6. The first of the two will be satisfied by assigning Z the value german (as translator\_speaks(english, german) is the first in the knowledge base), and now try to satisfy can\_translate(german, portuguese). This brings us back to stage 1, hence loops forever.
(c) The predicate \( \text{tnt}(X,Y,N) \) means “can translate from \( X \) to \( Y \), using at most \( N \) translators”. It is defined recursively. For the base case note that with at most one translator, we can translate whenever we \textit{can translate quickly}, or from any language to itself.

This implementation cannot loop forever – the first \( \text{tnt} \) goal will always have 6 as a third argument. Each recursive call will reduce that argument by 1, and once it reaches 0 the recursive clause will fail (at \( N>1 \)).

The last clause is read “we can translate from \( X \) to \( Y \) if we can translate from \( X \) to \( Y \) using at most 6 translators”. This is a correct definition since there are 6 translators in the firm. Clearly any bigger number would work (although it will take more time). As for smaller numbers – the longest path in the \textit{translator speaks} graph is of length 3. Therefore if a translation is possible, 3 translators will suffice. Consequently the 6 can be replaced by anything greater than or equal to 3 (but not less).