(1) Let \((\mathbb{Q}, <)\) be the structure with domain the set of rational numbers and the binary predicate 
< interpreted as the natural order on rationals.

(a) Write \((\forall x)(\exists y)(y < x)\) as a simple English sentence.

(b) An ordered set is called dense if between any two distinct members of the set there is a 
third. Write the proposition "The set of rational numbers is dense" in predicate logic.

(2) Give a resolution proof of the following propositional formulas (first convert their negation 
into clausal form).

(a) \(((P \rightarrow Q) \land (P \rightarrow (Q \rightarrow R))) \rightarrow (P \rightarrow R)\)

(b) \(((P \land Q) \lor (R \land \neg Q)) \rightarrow ((Q \rightarrow P) \land (\neg Q \rightarrow R))\)

(3) For each of the following formulas, first construct a tableaux proof, then give a resolution 
proof.

(a) \((\exists x)(\forall y)P(x, y) \rightarrow (\forall x)(\exists y)P(y, x)\)

(b) \((\forall x)(P(x, f(x)) \lor Q(g(x), x)) \rightarrow (\forall x)(\exists y)(P(x, y) \lor Q(y, x))\).

(4) Can the following sets of terms be unified? If there is a most general unifier, find it, otherwise 
explain why the terms cannot be unified. (\(a\) is a constant, \(x, y, z, w\) are variables, \(f, g, h\) are 
function symbols.)

(a) \(\{P(f(x), g(f(w)), x), P(y, g(z), g(a))\}\)

(b) \(\{f(x, g(a)), f(h(y), z), f(w, w)\}\)

(c) \(\{Q(x, f(x)), Q(g(y), y)\}\)
(5) Labeled binary trees are binary trees of which all leaves have labels. We represent a labeled leaf as `leaf(label)`, and a tree as `tree(left_subtree, right_subtree)` (recursively).

(a) Define a binary predicate `mirror` which succeeds when given two trees which are mirror images of one another. Invoking the predicate with a variable as a second argument should produce a mirror image of the tree in the first argument. For example:

```prolog
?- mirror(tree(tree(leaf(1),leaf(2)),leaf(3)), T).
T = tree(leaf(3),tree(leaf(2),leaf(1)))
yes
```

(b) Define a unary predicate `look_for_elephant` which searches for an elephant on a given tree. For example:

```prolog
?- look_for_elephant(tree(tree(leaf(cat),leaf(elephant)),leaf(whale))).
yes

?- look_for_elephant(tree(tree(leaf(cat),leaf(dog)),tree(leaf(bat),leaf(owl)))).
no
```