Exercise 1 (Proving with tactics).

Prove the following formulas using basic tactics for logical connectives. Do not use automation (`auto`, `tauto` or `firstorder`).

1. $(P \to Q) \to \neg Q \to \neg P$.
2. $P \to (P \to Q) \to Q$.
3. $P \to \neg\neg P$.
4. $P \lor \neg P \to \neg\neg P \to P$.
5. $\neg\neg(P \to Q) \to \neg\neg P \to \neg\neg Q$.
6. $(\neg\neg P \to \neg\neg Q) \to \neg\neg(P \to Q)$.
7. $\neg(P \land \neg P)$.
8. $(P \lor Q \to R) \to (P \to R) \land (Q \to R)$.
9. $(\exists x.P \land Rx) \leftrightarrow P \land \exists x.Rx$.
10. $P \lor (\forall x.Rx) \to \forall x.P \lor Rx$.
11. $(\exists x.Rx) \to \neg(\forall x.\neg Rx)$.
12. $\neg(P \lor Q) \leftrightarrow \neg P \land \neg Q$.

Exercise 2 (Induction on natural numbers).

Prove by induction the following facts about addition. Do not use `lia` or `omega`.

Hint. Use the `auto` tactic. It knows some basic facts about natural numbers.

1. Commutativity: `forall x y, x + y = y + x`.
2. Associativity: `forall x y z, (x + y) + z = x + (y + z)`.
3. Distributivity of multiplication: `forall x y z, x * (y + z) = x * y + x * z`.

   Hint. Use the previous two points and the tactic `replace t with t'` which replaces `t` with `t'` in the goal.

Exercise 3 (Induction on lists).

Consider the following function which reverses a list.

```
Fixpoint reverse {A} (l : list A) :=
  match l with
  | [] => []
  | h :: t => reverse t ++ [h]
  end.
```

This is *not* an efficient implementation of list reversal (quadratic complexity), but it is easier to reason with than the linear implementation from the first lecture.

Prove by induction the following facts about `reverse`.

1. `forall l1 l2 : list A, reverse (l1 ++ l2) = reverse l2 ++ reverse l1.`

   Hint. Use `SearchRewrite` to find helper lemmas in Coq's standard library.

2. `forall l : list A, reverse (reverse l) = l.`

3. `forall l : list A, reverse l = List.rev l.`

   Is it possible to prove `reverse = List.rev`?