Exercise 1 (Tail-recursive reverse).

In this exercise we prove some properties of the tail-recursive list reversal function from the first lecture. To make the task easier, we move the recursive helper function to a separate definition.

```
Fixpoint itrev {A} (lst acc : list A) :=
  match lst with
  | [] => acc
  | h :: t => itrev t (h :: acc)
  end.

Definition rev {A} (lst : list A) := itrev lst [].
```

Prove by induction the following facts about `rev`.

1. `forall l1 l2 : list A, rev (l1 ++ l2) = rev l2 ++ rev l1.`
2. `forall l : list A, rev (rev l) = l.`
3. `forall l : list A, rev l = List.rev l.`

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

Hint. You need to formulate an appropriate helper lemma about `itrev`. Recall the induction heuristics from the last lecture.

Exercise 2 (Palindromes).

Define an inductive predicate

```
Inductive Palindrome {A : Set} : list A -> Prop := ...
```

such that `Palindrome l` is provable iff the list `l` is a palindrome, i.e., it is equal to its own reversal. Prove:

1. `forall A (l : list A), Palindrome l -> List.rev l = l.`

*2. `forall A (l : list A), List.rev l = l -> Palindrome l.`

***Exercise 3** (Extensionality).

1. Show that predicate extensionality implies propositional extensionality.

   Hint. For variables P, Q, the equality $P = Q$ is equivalent to

   $$(\lambda x : \texttt{bool}.P)\texttt{true} = (\lambda x : \texttt{bool}.Q)\texttt{true}.$$

2. Show that propositional extensionality and functional extensionality together imply predicate exensionality.

3. Show that propositional extensionality and functional extensionality together imply the following statement:

   $$\forall AB : \text{Type}.\forall R_1 R_2 : A \to B \to \text{Prop}.(\forall xy.R_1 xy \leftrightarrow R_2 xy) \to R_1 = R_2.$$