Exercise 1 (Transitive-reflexive closure).

Define an inductive predicate

```
Inductive Star {A} (R : A -> A -> Prop) : A -> A -> Prop := ...
```

such that `Star R` is the transitive-reflexive closure of `R`. Prove:

```
forall R x y, Star R x y <-> TransitiveReflexiveClosure R x y
```

where `TransitiveReflexiveClosure` is an impredicative higher-order definition of the transitive-reflexive closure of a binary relation (see the lecture on higher-order logic).

Exercise 2 (Permutations).

Coq's standard library includes a `Permutation` inductive predicate which expresses that two lists are permutations of each other. Execute the Coq commands

```
Require Import Sorting.Permutation.
Print Permutation.
```

and study the definition of `Permutation`. Convince yourself that this inductive predicate indeed captures the notion of list permutation.

Prove the following properties of the `Permutation` inductive predicate.

1. If `Permutation l1 l2` then `List.length l1 = List.length l2`.
2. If `Permutation l1 l2` then `Permutation l2 l1`.
3. If `Permutation l1 l2` and `List.Forall P l1` then `List.Forall P l2`.

   Hint. Use `inversion` or `inversion_clear`.

Exercise 3 (Reversal is a permutation).

Prove `Permutation (rev l) l` by induction on `l`, where `rev` is the tail-recursive reverse function from the previous exercise sheet 06. Do not use the lemmas proven in exercise sheet 06.

Hint. You need to formulate a helper lemma about `itrev`.