Exercise 1 (The `take` and `drop` functions).

Consider the two functions below.

```
Fixpoint take {A} (n : nat) (l : list A) :=
  match l with
  | [] => []
  | x :: t =>
    match n with
    | 0 => []
    | S m => x :: take m t
    end
  end.

Fixpoint drop {A} (n : nat) (l : list A) :=
  match l with
  | [] => []
  | x :: t =>
    match n with
    | 0 => l
    | S m => drop m t
    end
  end.
```

Prove the following facts about `take` and `drop`.

1. `forall (l : list A) n, List.length (take n l) = min n (List.length l).`

   Hint. Do induction on `l` followed by case analysis on `n`.

2. ```
   forall (l1 l2 : list A) n,
       n < List.length l1 -> take n (l1 ++ l2) = take n l1.
   ```

   Hint. Use the `lia` tactic to solve arithmetic subgoals.

3. ```
   forall (l1 l2 : list A) n,
       n >= List.length l1 ->
       take n (l1 ++ l2) = l1 ++ take (n - List.length l1) l2.
   ```

4. `forall (l : list A) n, take n l ++ drop n l = l.`

Exercise 2 (Higher-order logic).

Define the equivalence closure `EC`(R) of a binary relation R as the intersection of all equivalence relations including R. Prove that `EC`(R) is the least equivalence relation including R.

Hint. Use the `firstorder` tactic to automate first-order logical reasoning.

Exercise 3 (Higher-order encodings of logical connectives)**.**

Show that the higher-order encodings of logical connectives from the lecture are equivalent to the corresponding connectives in Coq. More precisely, prove the following.

1. $\forall P P \leftrightarrow \bot$.
2. $\forall P((A \to B \to P) \to P) \leftrightarrow A \wedge B$.
3. $\forall P((A \to P) \to (B \to P) \to P) \leftrightarrow A \vee B$.
4. $\forall P(\forall x(Rx \to P) \to P) \leftrightarrow \exists x Rx$.
5. $\forall R(Rx \to Ry) \leftrightarrow x = y$.