Exercise 1 (Tail-recursive sum)**.** Consider the following two list summation functions, one tail-recursive and one not.

```
Fixpoint sum (l : list nat) : nat :=
  match l with
  | [] => 0
  | x :: xs => x + sum xs
  end.

Fixpoint itsum (l : list nat) (acc : nat) : nat :=
  match l with
  | [] => acc
  | x :: xs => itsum xs (x + acc)
  end.

Definition sum' (l : list nat) : nat :=
  itsum l 0.
```

Prove that:

```
forall l, sum l = sum' l
```

Hint. You need to formulate an appropriate helper lemma about `itsum`.

Exercise 2 (Universes, predicativity and impredicativity)**.**

1. Write two identity functions:

   ```
   id {T : Type} : T -> T
   pid {T : Prop} : T -> T
   ```

   Which of the following terms are well-typed and why?

   - `id (@id)`
   - `pid (@pid)`
   - `id Set`
   - `pid Set`
   - `id Type`
   - `pid Type`
   - `id 0`
   - `pid 0`

2. Write a function

   ```
   arrows : nat -> Set -> Set
   ```

   such that

```
arrows n T = T -> ... -> T -> T
```

where `T` occurs $n+1$ times (i.e., the result is a function type with n arguments).

Is `arrows 3 (arrows 3 nat)` well-typed? Why? What if we change the type of `arrows` to `nat -> Set -> Type`?

***Exercise 3** (Proof irrelevance)**.**

1. Show that propositional extensionality implies proof irrelevance.

   Hint. If we have a proof of P, then we can show $P \leftrightarrow \top$. In Coq, `True` is an inductive type with one constructor.

2. Show that predicate extensionality implies proof irrelevance.