Challenge

Spare Time Teaching

March 16, 2014

You may not add parameters or change the output.

Problem

Implement a `merge` function in Coq, which merges two sorted lists into one sorted list. And prove two small lemmas.

Example

```
Require Import Arith List.

Inductive is_sorted : list ℕ → Prop :=
  | s_nil :
    is_sorted nil
  | s_single : ∀ n,
    is_sorted (n :: nil)
  | s_cons : ∀ hd1 hd2 tl,
    hd1 ≤ hd2 →
    is_sorted (hd2 :: tl) →
    is_sorted (hd1 :: hd2 :: tl).

Fixpoint insert x xs :=
  match xs with
    | nil ⇒ x :: nil
    | x' :: xs' ⇒
      if leb x x' then
        x :: x' :: xs'
      else
        x' :: insert x xs'
  end.

(*** implement merge here ***)

Lemma insert_merge : ∀ x xs,
  is_sorted xs →
```

```
  insert x xs = merge (x :: nil) xs.
Proof.
  (* put proof here *)
Qed.

Lemma merge_correct : ∀ xs ys,
  is_sorted xs → is_sorted ys →
  is_sorted (merge xs ys).
Proof.
  (* put proof here *)
Qed.
```