```
(*  apl-2013-06-24b.v
 *
 * Programmer: Mayer Goldberg, 2013
 *)

Require Import Arith.
Require Import Setoid.

Fixpoint fact n :=
  match n with
    | 0 ⇒ 1
    | S p ⇒ (S p) × fact p
  end.

Eval cbv in fact 5.

(*    = 120
     : nat
 *)

Eval lazy in fact 5.

(*    = 120
     : nat
 *)

Eval cbv in (fact 5 + fact 6).

Fixpoint factI n r :=
  match n with
    | 0 ⇒ r
    | S p ⇒ factI p (n × r)
  end.

Eval cbv in factI 5 1.

(*    = 120
     : nat
 *)

Lemma LfactI: ∀ n r, factI (S n) r = factI n (S n × r).
Proof.
  intros n r.
  reflexivity.
Qed.

Lemma LfactII: ∀ n r s, factI n (r × s) = r × factI n s.
Proof.
  induction n.
  induction r.
  intro s.
```

```
  repeat rewrite mult_0_l; reflexivity.
  unfold factI; reflexivity.
  induction r.
  intro s.

  repeat rewrite mult_0_l.
  rewrite LfactI.
  rewrite mult_comm.
  rewrite IHn.
  rewrite mult_0_l; reflexivity.

  intro s.
  repeat rewrite LfactI.
  replace (S n × (S r × s)) with (S r × S n × s).
  rewrite IHn.
  rewrite IHn.
  rewrite → ? mult_assoc.

  reflexivity.
  rewrite mult_assoc.
  replace (S r × S n) with (S n × S r).
  reflexivity.
  rewrite mult_comm.
  reflexivity.
Qed.

Lemma LfactIII: ∀ n r, factI (S n) r = (S n) × factI n r.
Proof.
  induction n.
  unfold factI.
  reflexivity.

  intro r.
  rewrite LfactI.
  rewrite LfactII.
  reflexivity.
Qed.

Theorem Tfact: ∀ n, fact n = factI n 1.
Proof.
  induction n.

  (*  fact 0 = factI 0 1 *)
  reflexivity.

  (*
  n : nat
  IHn : fact n = factI n 1
```

```
  ============================
   fact (S n) = factI (S n) 1
   *)
  unfold fact; fold fact.
  rewrite LfactIII.
  rewrite <- IHn; reflexivity.
Qed.

Fixpoint fibR n :=
  match n with
    | 0 => 0
    | 1 => 1
    | S((S k) as p) => fibR p + fibR k
  end.

Fixpoint fibI n a b :=
  match n with
    | 0 => a
    | S p => fibI p b (a + b)
  end.

Eval cbv in fibR 10.
Eval cbv in fibI 10 0 1.

Lemma LunfoldFibR: ∀ n, fibR (S (S n)) = fibR (S n) + fibR n.
Proof.
  reflexivity.
Qed.

Lemma LunfoldFibI: ∀ n a b, fibI (S n) a b = fibI n b (a + b).
Proof.
  reflexivity.
Qed.

Lemma LunfoldBoth: ∀ n a b, fibI (S n) a b = fibR (S n) × b + (fibR n) × a.
Proof.
  induction n.
  intros a b.

  unfold fibI; unfold fibR.
  rewrite mult_1_l, mult_0_l, plus_0_r; reflexivity.

  (*
  n : nat
  IHn : forall a b : nat, fibI (S n) a b = fibR (S n) * b + fibR n * a
  ============================
   forall a b : nat, fibI (S (S n)) a b = fibR (S (S n)) * b + fibR (S n) * a
   *)
```

```
  intros a b.
  rewrite LunfoldFibI.
  rewrite IHn.
  rewrite LunfoldFibR.
  rewrite mult_plus_distr_l.
  rewrite mult_plus_distr_r.
  rewrite 2 (plus_comm _ (fibR n × b)).
  rewrite (plus_comm (fibR (S n) × a)).
  rewrite plus_assoc.
  reflexivity.
Qed.

Theorem Tfib: ∀ n : nat, fibR n = fibI n 0 1.
Proof.
  destruct n.
  reflexivity.

  (*  fibR (S n) = fibI (S n) 0 1 *)
  rewrite LunfoldBoth.
  rewrite mult_0_r.
  rewrite mult_1_r.
  rewrite plus_0_r.
  reflexivity.
Qed.
```