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

Require Import Arith.

Theorem our_plus_0_l: ∀ a, 0 + a = a.
Proof.
  intro a.
  unfold plus; fold plus; reflexivity.
Qed.

Theorem our_plus_0_r: ∀ a, a + 0 = a.
Proof.
  induction a.

  reflexivity.

  unfold plus; fold plus.
  rewrite IHa; reflexivity.
Qed.

Lemma L1: ∀ a b, S a + b = S (a + b).
Proof.
  fold plus; unfold plus; reflexivity.
Qed.

Lemma L2: ∀ a b, a + S b = S (a + b).
Proof.
  induction a.
  induction b.
  reflexivity.

  repeat rewrite our_plus_0_l; reflexivity.

  induction b.

  rewrite our_plus_0_r; rewrite L1.

  rewrite IHa; rewrite our_plus_0_r; reflexivity.

  rewrite L1; rewrite IHa; rewrite L1; reflexivity.
Qed.

Theorem our_plus_comm: ∀ a b, a + b = b + a.
Proof.
  induction a.
  intro b.
  rewrite our_plus_0_l, our_plus_0_r; reflexivity.

  induction b.
```

```
  rewrite our_plus_0_l; rewrite our_plus_0_r; reflexivity.
  repeat rewrite L1.
  repeat rewrite L2.
  rewrite IHa.
  reflexivity.
Qed.

Theorem our_mult_0_l: ∀ a, 0 × a = 0.
Proof.
  intro a.
  unfold mult; reflexivity.
Qed.

Theorem our_mult_0_r: ∀ a, a × 0 = 0.
Proof.
  induction a.
  unfold mult; reflexivity.
  unfold mult; fold mult.
  rewrite our_plus_0_l.
  exact IHa.
Qed.

Theorem our_mult_1_l: ∀ a, 1 × a = a.
Proof.
  intro a.
  unfold mult.
  rewrite our_plus_0_r; reflexivity.
Qed.

Theorem our_mult_1_r: ∀ a, a × 1 = a.
Proof.
  induction a.
  rewrite our_mult_0_l; reflexivity.
  unfold mult; fold mult.
  unfold plus.
  rewrite IHa.
  reflexivity.
Qed.

Theorem our_plus_assoc: ∀ a b c, a + (b + c) = a + b + c.
Proof.
  induction a, b, c.

  reflexivity.

  repeat rewrite our_plus_0_l; reflexivity.

  repeat rewrite our_plus_0_l; repeat rewrite our_plus_0_r; reflexivity.
```

```
  repeat rewrite our_plus_0_l; reflexivity.
  repeat rewrite our_plus_0_r; reflexivity.
  rewrite our_plus_0_l, our_plus_0_r; reflexivity.
  repeat rewrite our_plus_0_r; reflexivity.
  repeat rewrite L1; repeat rewrite L2; rewrite L1; rewrite IHa; reflexivity.
Qed.
Lemma L3: ∀ a b, S a × b = b + a × b.
Proof.
  intros a b.
  unfold mult; fold mult; reflexivity.
Qed.
Lemma L4: ∀ a b, a × S b = a + a × b.
Proof.
  induction a.
  intro b.
  repeat rewrite our_mult_0_l.
  rewrite our_plus_0_l; reflexivity.
  induction b.
  rewrite our_mult_1_r, our_mult_0_r, our_plus_0_r; reflexivity.
  repeat rewrite L3.
  repeat rewrite IHa.
  repeat rewrite our_plus_assoc.
  replace (S (S b) + a) with (S a + S b).
  reflexivity.
  repeat rewrite L1.
  rewrite L2, our_plus_comm; reflexivity.
Qed.
Theorem our_mult_comm: ∀ a b, a × b = b × a.
Proof.
  induction a.
  intro b.
  rewrite our_mult_0_l, our_mult_0_r; reflexivity.
  induction b.
  rewrite our_mult_0_l, our_mult_0_r; reflexivity.
  repeat rewrite L3.
  repeat rewrite L4.
  repeat rewrite our_plus_assoc.
  rewrite ← IHa.
  repeat rewrite L1.
```

```
  replace (b + a) with (a + b).
  reflexivity.
  apply our_plus_comm.
Qed.

Theorem our_mult_plus_distr_r: ∀ a b c, (a + b) × c = a × c + b × c.
Proof.
  induction a, b, c.
  repeat rewrite our_mult_0_r; reflexivity.
  repeat rewrite our_mult_0_l; reflexivity.
  repeat rewrite our_mult_0_r; reflexivity.
  repeat rewrite our_mult_0_l; repeat rewrite our_plus_0_l; reflexivity.
  repeat rewrite our_mult_0_r; reflexivity.
  rewrite our_plus_0_r, our_mult_0_l, our_plus_0_r; reflexivity.
  repeat rewrite our_mult_0_r; reflexivity.
  rewrite L1.
  rewrite L2.
  repeat rewrite L3.
  repeat rewrite L4.
  rewrite IHa.
  repeat rewrite our_plus_assoc.
  repeat rewrite L1.
  repeat rewrite L2.
  repeat rewrite L1.
  rewrite (our_plus_comm (c + a + a × c) c).
  repeat rewrite our_plus_assoc.
  replace ((c + c + a) + b + (a × c)) with ((c + c + a) + (a × c) + b).
  reflexivity.
  repeat rewrite ← our_plus_assoc.
  rewrite (our_plus_comm (a × c) b).
  reflexivity.
Qed.

Theorem our_mult_plus_distr_l: ∀ a b c, a × (b + c) = a × b + a × c.
Proof.
  intros a b c.
  rewrite our_mult_comm.
  rewrite our_mult_plus_distr_r.
  rewrite our_mult_comm at 1.
  rewrite (our_mult_comm c a).
  reflexivity.
Qed.

Lemma acPbc: ∀ a b c, a = b → a + c = b + c.
Proof.
```

```
  intros a b c H.
  rewrite H; reflexivity.
Qed.

Theorem our_mult_assoc: ∀ a b c, a × (b × c) = a × b × c.
Proof.
  induction a, b, c.

  repeat rewrite our_mult_0_r; reflexivity.
  repeat rewrite our_mult_0_l, our_mult_0_r; reflexivity.
  repeat rewrite our_mult_0_l, our_mult_0_r; reflexivity.
  repeat rewrite our_mult_0_l; reflexivity.
  repeat rewrite our_mult_0_r; reflexivity.
  rewrite our_mult_0_l, our_mult_0_r; reflexivity.
  repeat rewrite our_mult_0_r; reflexivity.
  repeat rewrite L3, L4.
  repeat rewrite our_plus_assoc.
  repeat rewrite our_mult_plus_distr_r.
  repeat rewrite our_mult_plus_distr_l.
  repeat rewrite our_plus_assoc.
  rewrite IHa.
  rewrite (acPbc (S b + c + b × c + a × S b + a × c) (S b + a + a × b + S b × c +
a × c) (a × b × c)).
  reflexivity.
  rewrite (acPbc (S b + c + b × c + a × S b) (S b + a + a × b + S b × c) (a × c)).
  reflexivity.
  rewrite L3.
  rewrite L4.
  repeat rewrite ← our_plus_assoc.
  repeat rewrite (our_plus_comm (S b) _).
  rewrite (acPbc (c + (b × c + (a + a × b))) (a + (a × b + (c + b × c))) (S b)).
  reflexivity.
  rewrite (our_plus_comm (b × c) (a + a × b)).
  repeat rewrite our_plus_assoc.
  rewrite (acPbc (c + a + a × b) (a + a × b + c) (b × c)).
  reflexivity.
  rewrite ← (our_plus_assoc a (a × b) c).
  rewrite (our_plus_comm (a × b) c).
  rewrite our_plus_assoc.
  rewrite (our_plus_comm a c).
  reflexivity.
Qed.
```