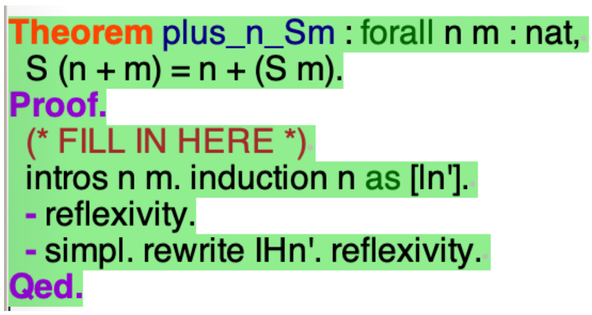
1.

```
Theorem plus_n_Sm : forall n m : nat,
  S (n + m) = n + (S m).
Proof.
  (* FILL IN HERE *)
  intros n m. induction n as [|n'].
  - reflexivity.
  - simpl. rewrite IHn'. reflexivity.
Qed.
```

分析：

对n做归纳即可。

运行结果：



```
Theorem plus_n_Sm : forall n m : nat,
  S (n + m) = n + (S m).
Proof.
  (* FILL IN HERE *)
  intros n m. induction n as [|n'].
  - reflexivity.
  - simpl. rewrite IHn'. reflexivity.
Qed.
```

2.

```
(** **** Exercise: 3 stars, standard, especially useful (mul_comm)

    Use [assert] to help prove [add_shuffle3].  You don't need to
    use induction yet. *)

Theorem add_shuffle3 : forall n m p : nat,
  n + (m + p) = m + (n + p).
Proof.
  intros n m p.
  assert (H: n + (m + p) = (n + m) + p).
    { rewrite add_assoc. reflexivity. }
  rewrite H.
  assert (H': m + (n + p) = (m + n) + p).
    { rewrite add_assoc. reflexivity. }
  rewrite H'.
  assert (H'': n + m = m + n).
    { rewrite add_comm. reflexivity. }
  rewrite H''.
  reflexivity.
  Qed.
```

分析：

根据提示进行多次assert即可证明。

运行结果：

```
Theorem add_shuffle3 : forall n m p : nat,
  n + (m + p) = m + (n + p).
Proof.
  intros n m p.
  assert (H: n + (m + p) = (n + m) + p).
    { rewrite add_assoc. reflexivity. }
  rewrite H.
  assert (H': m + (n + p) = (m + n) + p).
    { rewrite add_assoc. reflexivity. }
  rewrite H'.
  assert (H'': n + m = m + n).
    { rewrite add_comm. reflexivity. }
  rewrite H''.
  reflexivity.
  Qed.
```

3.

```
Theorem mul_n_Sm : forall n m : nat,  n + n * m = n * S m.
Proof.
  intros n m. induction n.
  - reflexivity.
  - simpl. rewrite <- IHn. rewrite add_assoc.
    assert (H': m + (n + n * m) = m + n + n * m).
      { rewrite add_assoc. reflexivity. }
    rewrite H'.
    assert (H'': n + m = m + n).
      { rewrite add_comm. reflexivity. }
    rewrite H''.
    reflexivity.
Qed.
```

分析：

对n做归纳，通过assert逐步证明。

运行结果：

```
Theorem mul_n_Sm : forall n m : nat,  n + n * m = n * S m.
Proof.
  intros n m. induction n.
  - reflexivity.
  - simpl. rewrite <- IHn. rewrite add_assoc.
    assert (H': m + (n + n * m) = m + n + n * m).
      { rewrite add_assoc. reflexivity. }
    rewrite H'.
    assert (H'': n + m = m + n).
      { rewrite add_comm. reflexivity. }
    rewrite H''.
    reflexivity.
Qed.
```

4.

```
Theorem mul_n_0 : forall n : nat, n * 0 = 0.
Proof.
  intros n. induction n.
  - reflexivity.
  - simpl. rewrite IHn. reflexivity.
Qed.

(** Now prove commutativity of multiplication.  You will probably want
    to look for (or define and prove) a "helper" theorem to be used in
    the proof of this one. Hint: what is [n * (1 + k)]? *)

Theorem mul_comm : forall m n : nat,
  m * n = n * m.
Proof.
  intros m n.
  induction m.
  - simpl. rewrite mul_n_0. reflexivity.
  - simpl. rewrite <- mul_n_Sm. rewrite IHm. reflexivity.
Qed.
```

分析：

根据提示，我们要先证明一个引理，也就是上一题的证明；并且在对m做归纳之后，发现base case还需要证明另一个引理`n * 0 = 0`。

运行结果：

```
Theorem mul_n_0 : forall n : nat, n * 0 = 0.
Proof.
  intros n. induction n.
  - reflexivity.
  - simpl. rewrite IHn. reflexivity.
Qed.


Theorem mul_comm : forall m n : nat,
  m * n = n * m.
Proof.
  intros m n.
  induction m.
  - simpl. rewrite mul_n_0. reflexivity.
  - simpl. rewrite <- mul_n_Sm. rewrite IHm. reflexivity.
Qed.
```