

1.

```
1 Theorem plus_n_Sm : forall n m : nat,  
2   S (n + m) = n + (S m).  
3 Proof.  
4   (* FILL IN HERE *)  
5   intros n m. induction n as [|n'].  
6   - reflexivity.  
7   - simpl. rewrite IHn'. reflexivity.  
8 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.

```
1  (** **** Exercise: 3 stars, standard, especially useful (mul_comm)  
2  
3      Use [assert] to help prove [add_shuffle3]. You don't need to  
4      use induction yet. *)  
5  
6 Theorem add_shuffle3 : forall n m p : nat,  
7   n + (m + p) = m + (n + p).  
8 Proof.  
9   intros n m p.  
10  assert (H: n + (m + p) = (n + m) + p).  
11    { rewrite add_assoc. reflexivity. }  
12  rewrite H.  
13  assert (H': m + (n + p) = (m + n) + p).  
14    { rewrite add_assoc. reflexivity. }  
15  rewrite H'.  
16  assert (H'': n + m = m + n).  
17    { rewrite add_comm. reflexivity. }  
18  rewrite H''.  
19  reflexivity.  
20 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.

```
1 Theorem mul_n_Sm : forall n m : nat, n + n * m = n * S m.  
2 Proof.  
3   intros n m. induction n.  
4   - reflexivity.  
5   - simpl. rewrite <- IHn. rewrite add_assoc.  
6     assert (H': m + (n + n * m) = m + n + n * m).  
7     { rewrite add_assoc. reflexivity. }  
8     rewrite H'.  
9     assert (H'': n + m = m + n).  
10    { rewrite add_comm. reflexivity. }  
11    rewrite H''.  
12    reflexivity.  
13 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.

```

1 Theorem mul_n_0 : forall n : nat, n * 0 = 0.
2 Proof.
3   intros n. induction n.
4   - reflexivity.
5   - simpl. rewrite IHn. reflexivity.
6 Qed.
7
8 (** Now prove commutativity of multiplication. You will probably want
9     to look for (or define and prove) a "helper" theorem to be used in
10    the proof of this one. Hint: what is [n * (1 + k)]? *)
11
12 Theorem mul_comm : forall m n : nat,
13   m * n = n * m.
14 Proof.
15   intros m n.
16   induction m.
17   - simpl. rewrite mul_n_0. reflexivity.
18   - simpl. rewrite <- mul_n_Sm. rewrite IHm. reflexivity.
19 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.