

1.

```
1  (** **** Exercise: 2 stars, standard (ev_sum) *)
2  Theorem ev_sum : forall n m, ev n -> ev m -> ev (n + m).
3  Proof.
4    intros. induction H as [ | n' H' IH].
5    - simpl. apply H0.
6    - simpl. apply ev_SS. apply IH.
7    Qed.
```

```
(** **** Exercise: 2 stars, standard (ev_sum) *)
Theorem ev_sum : forall n m, ev n -> ev m -> ev (n + m).
Proof.
  intros. induction H as [ | n' H' IH].
  - simpl. apply H0.
  - simpl. apply ev_SS. apply IH.
  Qed.
```

2.

```
1  (** **** Exercise: 5 stars, standard, optional (le_and_lt_facts) *)
2  Lemma le_trans : forall m n o, m <= n -> n <= o -> m <= o.
3  Proof.
4    intros. induction H0 as [ | n' H' IH].
5    - apply H.
6    - apply le_S. apply IH.
7    Qed.
```

```
Lemma le_trans : forall m n o, m <= n -> n <= o -> m <= o.
Proof.
  intros. induction H0 as [ | n' H' IH].
  - apply H.
  - apply le_S. apply IH.
  Qed.
```

3.

```
1 Theorem App_ : forall n m o p : nat, [n;m;o;p] = [n;m]++[o;p].
2 Proof. intros. reflexivity. Qed.
3
4 Example reg_exp_ex3 : [0;1;0;1] =~ Star (App (Char 0) (Char 1)).
5 Proof. rewrite -> App_.
6 apply MStarApp.
7 * apply (MApp [0] _ [1]).
8   + apply MChar.
9   + apply MChar.
10 * rewrite <- (app_nil_r _ [0;1]).
11   apply MStarApp.
12   + apply (MApp [0] _ [1]).
13     ++ apply MChar.
14     ++ apply MChar.
15   + apply MStar0.
16 Qed.
```

Theorem App_ : forall n m o p : nat, [n;m;o;p] = [n;m]++[o;p].
Proof. intros. reflexivity. Qed.

Example reg_exp_ex3 : [0;1;0;1] =~ Star (App (Char 0) (Char 1)).
Proof. rewrite -> App_.
apply MStarApp.
* apply (MApp [0] _ [1]).
 + apply MChar.
 + apply MChar.
* rewrite <- (app_nil_r _ [0;1]).
 apply MStarApp.
 + apply (MApp [0] _ [1]).
 ++ apply MChar.
 ++ apply MChar.
 + apply MStar0.
Qed.

4.

```
1 Lemma MUnion' : forall T (s : list T) (rel re2 : reg_exp T),
2   s =~ rel \/ s =~ re2 ->
3   s =~ Union rel re2.
4 Proof.
5   intros. destruct H.
6   - apply MUnionL. apply H.
7   - apply MUnionR. apply H.
8 Qed.
```

```
Lemma MUnion' : forall T (s : list T) (rel re2 : reg_exp T),
  s =~ rel \/ s =~ re2 ->
  s =~ Union rel re2.
Proof.
  intros. destruct H.
  - apply MUnionL. apply H.
  - apply MUnionR. apply H.
Qed.
```