Mini-Test 1

Name: ______________________________ Email: ____________________

1. (50 points) Check the correct answer.
   (a) (10 points) You can conclude proposition `2 <> 4` using **`reflexivity`**.
       ◯ True
       ◯ False
   (b) (10 points) The proof of an implication `P → Q` is a function that uses a proof of the proposition `P` to produce a proof of the proposition `Q`.
       ◯ True
       ◯ False
   (c) (10 points) If `E` has type `Nat.eqb m n = true`, then `E` can be applied to a goal `m = n`.
       ◯ True
       ◯ False
   (d) (10 points) If `X` is an inductively defined type or proposition with no constructors and `foo : X`, then **`destruct`** `foo` will finish any proof or subgoal.
       ◯ True
       ◯ False
   (e) (10 points) The type **`Inductive`** `foo := | bar: foo → foo.` is an invalid type definition in Coq.
       ◯ True
       ◯ False

2. (10 points) Give the type of each of the following Coq expressions, or write "ill typed" if an expression does not have a type.
   (a) (5 points) **`forall`** `(x : nat) (y :` **`Prop`**`), x → y`

   (b) (5 points) **`forall`** `(X Y :` **`Prop`**`), X → Y`

3. (20 points) For each of the following propositions, check "not provable" if it is not provable (in Coq's core logic, without additional axioms), "induction" if it is provable only using induction, or "easy" if it is provable without using induction and without additional lemmas.
   (a) (4 points) **`exists`** `s, In 3 (s ++ [1;2;3])`
       ◯ Easy
       ◯ Induction
       ◯ Not Provable
   (b) (4 points) **`forall`** `s, In 3 (s ++ [1;2;3])`
       ◯ Easy
       ◯ Induction

◯ Not Provable

(c) (4 points) `forall n, n = S n`

◯ Easy
◯ Induction
◯ Not Provable

(d) (4 points) `forall n, n + 0 = n`

◯ Easy
◯ Induction
◯ Not Provable

(e) (4 points) `forall {A:Type} (l:list A), l = [] \/ exists x l', l = x :: l'`

◯ Easy
◯ Induction
◯ Not Provable

4. (20 points) Complete each proof. Your proof cannot use **auto** nor **intuition**.

(a)
```
P, Q : Prop
H : P \/ Q
H0 : ~ Q
______________________________________(1/1)
P
```

(b) `forall (A:Type) (l:list A), l = [] → l = []`

(c) `forall (A:Type) (x:A), [x] = [x].`

(d)
```
H : P -> Q
H0 : P \/ ~ P
____________________________________(1/1)
~ P \/ Q
```