CIS 5000: Software Foundations Quiz 1

September 4, 2025

Name (printed): _____ PennKey: _____

Whenever you are asked whether something is provable, you should assume that no other facts or lemmas are available unless specified otherwise.

1 Recall the function `lower_letter` defined in `Basics.v` and reproduced on the handout sheet. Below it on the handout is a definition of a function `higher_letter` that *raises* a letter grade.

Now consider the following theorem statement:

```
Theorem higher_lower_equal:
  forall (l : letter), higher_letter (lower_letter l) = l.
```

Is this theorem provable (check one)? ☐ Yes ☒ No

If Yes, which tactics would to prove it (check all that apply)?

☐ `destruct` ☐ `rewrite` ☐ `reflexivity` ☐ `induction`

If No, which values for `l` provide counterexamples (check all that apply)? ☐ A ☐ B ☐ C ☐ D ☒ F

2 The definition of `plus` (`+`) is given on the handout.

Given the following proof state:

```
n : nat
============================
((S (S n)) + 1) = ((S n) + 2)
```

What does the goal look like after `simpl` is used?

☒ `(S (S (n + 1))) = (S (n + 2))`

☐ `(S (S (S n))) = (S (S (S n)))`

☐ `(2 + (n + 1)) = ((1 + n) + 2)`

☐ It does not "look like" anything, as `simpl` completely solves the goal.

3 Now consider an alternative definition of `plus` called `altplus`, defined on the handout.

Which tactics are **required**, in addition to `simpl`, to prove the following lemmas (check one)?

(a) `Lemma altplus_0_n: forall n, altplus 0 n = n.`

☐ `intros`.

☐ `intros` and `reflexivity`.

☒ `intros`, `rewrite`, `induction`, and `reflexivity`.

☐ We cannot prove it with any of the above.

(b) `Lemma altplus_n_0: forall n, altplus n 0 = n.`

☐ `intros`.

☒ `intros` and `reflexivity`.

☐ `intros`, `rewrite`, `induction`, and `reflexivity`.

☐ We cannot prove it with any of the above.

4 Consider the following lemma (`altplus` was defined in the previous question):

```
Lemma plus_equiv: forall n m, plus n m = altplus n m.
```

Which tactic can be used to prove it (check one)?

☐ `destruct` ☐ `induction n` ☐ `induction m` ☒ None of these

Now, assume that you have access to the following lemma about `altplus`:

```
Lemma altplus_comm: forall m n, altplus n m = altplus m n.
```

Which tactic can be used to prove `plus_equiv` (check one)?

☐ `destruct` ☒ `induction n` ☐ `induction m` ☐ None of these

5 For each of the following propositions, determine whether it requires induction to be proved. The definition of `leb` is repeated on the handout.

(a) `forall n, n + 0 = n`

☒ needs induction ☐ does not need induction ☐ not provable

(b) `forall n, 0 + n = n`

☐ needs induction ☒ does not need induction ☐ not provable

(c) `forall n, 0 leb 2 * n - n`

☐ needs induction ☒ does not need induction ☐ not provable

(d) `forall n m, m leb n = true -> n leb m = false`

☐ needs induction ☐ does not need induction ☒ not provable

(e) `forall n m q, q leb m = true -> m leb n = true -> q leb n = true`

☒ needs induction ☐ does not need induction ☐ not provable

Reference

```
Definition lower_letter (l : letter) : letter :=
    match l with
    | A => B
    | B => C
    | C => D
    | D => F
    | F => F
    end.

Definition higher_letter (l : letter) : letter :=
    match l with
    | A => A
    | B => A
    | C => B
    | D => C
    | F => D
    end.

Fixpoint plus (n: nat) (m: nat): nat :=
  match n with
  | O => m
  | S n' => S (plus n' m)
  end.

Fixpoint altplus (n: nat) (m: nat): nat :=
  match m with
  | O => n
  | S m' => S (altplus n m')
  end.

Fixpoint leb (n m : nat) : bool :=
  match n with
  | O => true
  | S n' =>
      match m with
      | O => false
      | S m' => leb n' m'
      end
  end.

Notation "x <=? y" := (leb x y) (at level 70).
```