CIS 5000: Software Foundations Quiz 4 September 25, 2025

Name (printed): _____ Penn ID (8-digit #): _____

Please write your name as it is displayed on Gradescope. Indicate your answers with a cross mark ☒. To correct a mistake, fill in the incorrect box (like this: ■) and make a cross in the correct box.

1 A number is *prime* if its only divisors are itself and one. Which of the following Rocq statements are logically equivalent to the (false) informal proposition "All non-prime numbers are divisible by two"? Select all that apply.

(a) ☐
```
forall (n : Nat),
    ~(forall p q, p * q <> n) ->
    exists (r : Nat), n / 2 = r.
```

(b) ☐
```
forall (n : Nat),
    (exists p q, p * q = n /\ p <> 1 /\ q <> 1) ->
    exists (r : Nat), r * 2 = n.
```

(c) ☐
```
forall (n : Nat),
     (forall p q, p > 1 -> q > 1 -> p * q <> n) ->
     exists (r : Nat), r * 2 = n.
```

(d) ☐
```
forall (n : Nat),
    (exists (r : Nat), n = 2 * r) ->
    exists p q, p * q = n.
```

(e) ☐
```
forall (n : Nat),
    (exists p q, (S p) * (S q) = n) ->
    forall (r : Nat), S (r * 2) <> n.
```

Regardless of their equivalence to the original proposition, which of the above statements are *true*? Check all that apply.

☐ (a) ☐ (b) ☐ (c) ☐ (d) ☐ (e)

2 If the current goal is `R` and the context contains a hypothesis `H : P /\ Q /\ R`, which tactic should we use?

☐ `apply H.`

☐ `destruct H as [HP HQ HR].`

☐ `split.`

☐ `left.`

3 Consider the following theorem:

```
forall (A : Type) (P Q : A -> Prop),
  (exists x, P x) \/ (exists x, Q x) -> exists x, P x \/ Q x.
```

(a) Is this theorem provable in Rocq, using the tactics we have covered so far?

☐ Yes. ☐ No.

(b) If yes, which tactics would we use to prove it, besides `intros` and `apply`? Select all that apply.

☐ `exists` ☐ `destruct` ☐ `discriminate`

☐ `left` and/or `right` ☐ `unfold`

(c) If no, why not?

☐ Because it's false.

☐ Because we have not learned the necessary tactics yet.

☐ Because it's ill-formed.

☐ Because we need classical logic to prove it.

☐ None of the above.

4 What are the types of the following expressions?

(a) `fun n:nat => S (pred n)`

☐ `Prop` ☐ `Nat -> Nat`

☐ `Nat -> Prop` ☐ Not typeable.

☐ `forall n : Nat, Prop`

(b) `forall n:nat, S (pred n)`

☐ `Prop` ☐ `Nat -> Nat`

☐ `Nat -> Prop` ☐ Not typeable.

☐ `forall n : Nat, Prop`

5 Which of the following is *not* a proposition?

☐ `3 + 2 = 4`

☐ `3 + 2 =? 5`

☐ `((3+2) =? 4) = false`

☐ `forall n, (((3+2) =? n) = true) -> n = 5`

☐ All of these are propositions.