EECS 755 - Midterm Exam

Spring Semester 2022

March 31, 2022

Exercise 1 *Which of the following statements are true? (1pt each)*

1. The `induction` tactic implements proof-by-cases
2. The `Definition` construct can define a recursive function.
3. The `rewrite <- H` tactic applied with `H:A=B` will replace `A` with `B`.
4. The `unfold` tactic replaces a function with its definition
5. An enumeration is an `Inductive` type with no recursion.
6. The `reflexivity` tactic solves any goal of the form `A = A`.
7. The `simpl` or `intros` tactics get rid of universally quantified variables in the goal.
8. The `intros` command will transform the goal `x=y -> y=x` into the new goal `x=y` and adds `y=x` to the assumptions.
9. When using `Inductive` to define a type, the resulting constructors together create every value of a specified type.
10. A decision procedure determines if a term is true or false.
11. Coq functions may take types as arguments and produce types as results.
12. A value of `p` given the declaration `p:x<y` is a proof that `x<y`.
13. A transition system defines a finite collection of states and a transition function.
14. In The Adventures of Buckaroo Bonzai, Perfect Tommy is the bass player for The Hong Kong Cavaliers

Exercise 2 *In this problem you will define an inductive type for n-ary trees of natural numbers. Recall that an n-ary tree is a tree where nodes have a value and an arbitrary number of branches. A leaf node is a node with zero branches.*

1. *Define an* `Inductive` *type for n-ary integer trees.*
2. *Define a function* `search` *that will search an n-ary tree for some natural number and return* `option nat` *where the* `None` *constructor indicates not found and* `Some` *returns a natural number.*
3. *Give some n-ary tree,* `t`*, what proof goals will* `induction t` *generate?*
4. *Rework your definition to make your n-ary tree definition polymorphic.*
5. *Rework your* `search` *function to be polymorphic over your polymorphic tree. In other words,* `search` *should work no matter what type is in the tree. (This is tricky, be careful)*

Exercise 3 *Assume the following two inductive data type definitions:*

```
Inductive colors : Type :=
| red : nat -> colors
| yellow : nat -> colors
| green : nat -> colors.
```

```
Inductive stack : Type :=
| empty : stack
| push : nat -> stack -> stack.
```

1. *If a proof goal has the form* `s=s` *where* `s` *is a stack, what proof tactic wold you apply first? Describe the result.*
2. *If a proof goal has the form* `forall n s, push(n,s)<>empty` *what proof tactic would you apply first? Describe the result.*
3. *If a proof goal has the form* `p c` *where* `p` *is a property and* `c` *is a* `color`*, what proof tactic would you apply? Describe the result.*
4. *If a proof goal has the form* `p s` *where* `p` *is a property and* `s` *is a* `stack`*, what proof tactic would you apply? Describe the result.*
5. *Given a term of the form* `H:A->B=C`*, what is the difference between apply H and rewrite H?*
6. *Given* `s:stack` *what goals and assumptions are generated by* `induction s`*?*
7. *Given* `s:stack` *what goals and assumptions are generated by desruct s?*
8. *Does* `discriminate` *do anything with* `(push 2 s) = (push 3 s)`*? If so, what?*
9. *Does* `injection` *do anything with* `(push 2 s) = (push 3 s)`*? Describe the result.*
10. *Given* `c:colors`*, what is the difference between* `induction c` *and* `destruct c`*?*