CIS500 Final Review

April 29, 2013

What is the result of the following expression?

```
map (fun n => n * n) [1,2,3]
```

1. `[1,2,3]`
2. `[1,4,9]`
3. 3
4. 9
5. None of the above

What is the following expression equal to?

`A -> B -> C`

1. `A -> (B -> C)`
2. `(A -> B) -> C`
3. None of the above

What is the type of the following function?

```
Fixpoint fold A B (f : A -> B -> B) (b : B)
                  (l : list A) : B :=
  match l with
  | [] => b
  | a :: l' => f a (fold f b l')
  end.
```

1. `A -> B -> (A -> B -> B) -> B -> list A -> B`
2. `forall A B, f -> b -> l -> B`
3. `forall A B, A -> B -> B -> B -> list A -> B`
4. `forall A B, (A -> B -> B) -> B -> list A -> B`
5. Ill-typed

Suppose that `A : Type`, `P : A -> Prop` and `Q : A -> A -> Prop`. Are the following propositions logically equivalent? (i.e., `P1 <-> P2`)

```
forall a : A, P a -> forall b : A, Q a b
```

```
forall a b : A, P a -> Q a b
```

1. Yes
2. No

Suppose that `A : Type` and `Q : A -> A -> Prop`. Are the following propositions logically equivalent? (i.e., `P1 <-> P2`)

```
forall a : A, exists b : A, Q a b
```

```
exists b : A, forall a : A, Q a b
```

1. Yes
2. No

Which logic connective does the following declaration define?

```
Inductive R (A B : Prop) : Prop :=
| R_intro : A -> B -> R A B.
```

1. `or`
2. `and`
3. `exists`
4. `forall`
5. None of the above.

What is the type of the following expression?

```
fun A B (H : A \/ B) =>
  match H with
  | or_introl HA => or_intror HA
  | or_intror HB => or_introl HB
  end
```

1. `forall A B, A \/ B -> A /\ B`
2. `forall A B, A \/ B -> B \/ A`
3. `forall A B, A \/ B -> A \/ B`
4. None of the above

Define the "strictly less than" relation as a Coq inductive predicate.

Annotate the following program.

```
{True}
R ::= X;
Q ::= 0;
WHILE R >= D DO
  R ::= R - D;
  Q ::= Q + 1
END
{X = Q * D + R /\ R < D}
```

Find an invariant that can be used to prove the following specification holds.

```
{X = n}
Y ::= 1;
WHILE X > 0 DO
  Y ::= 2 * Y;
  X ::= X - 1
END
{Y = 2^n}
```

Find an invariant that can be used to prove the following specification holds.

```
{True}
Y ::= 0;
P ::= 0;
WHILE Y * Y <= X DO
  IF Y * Y = X THEN
    P ::= 1;
  ELSE
    SKIP
  FI;
  Y ::= Y + 1
END
{P = 0 /\ (forall k, k * k <> X) \/
 P = 1 /\ (exists k, k * k = X)}
```

Recall that the *type-safety* property states informally that *well-typed programs can't go wrong*. How do you formalize this statement in Coq?

Suppose that we add the following evaluation rule to the STLC:

```
--------------------------------
if true then t1 else t2 ==> true
```

Is `step` still deterministic?

1. Yes
2. No

Suppose that we add the following evaluation rule to the STLC:

```
--------------------------------
if true then t1 else t2 ==> true
```

Does progress still hold?

1. Yes
2. No

Suppose that we add the following evaluation rule to the STLC:

```
--------------------------------
if true then t1 else t2 ==> true
```

Does preservation still hold?

1. Yes
2. No

Suppose that we add the following evaluation rules:

```
     value v
----------------
true v ==> false

     value v
----------------
false v ==> true
```

Is `step` still deterministic?

1. Yes
2. No

Suppose that we add the following evaluation rules:

```
     value v
----------------
true v ==> false

     value v
----------------
false v ==> true
```

Does progress still hold?

1. Yes
2. No

Suppose that we add the following evaluation rules:

```
     value v
----------------
true v ==> false

     value v
----------------
false v ==> true
```

Does preservation still hold?

1. Yes
2. No

Does the following proposition hold?

```
exists T, empty |- (\p : T. p.fst (p.snd 42)) : T -> A
```

1. Yes
2. No

Does the following proposition hold?

```
exists T, empty |- fix (\n : Nat. pred n) : T
```

1. Yes
2. No

Does the following proposition hold?

```
exists T T',
  y : T' |- y (if true then true else y) : T
```

1. Yes
2. No

Give an informal proof of the following theorem about the STLC with booleans:

```
forall Gamma t T1 T2,
  Gamma |- t : T1 ->
  Gamma |- t : T2 ->
  T1 = T2
```