CS 421 --- Type Classes

Manager	Keeps team on track	
Recorder	Records decisions	
Reporter	Reports to class	
Reflector	Assesses team performance	

Code

Consider the following code. As a team, review the code and be sure everyone understands what is happening. The questions below may help with that.

```haskell
{-# LANGUAGE FlexibleInstances #-}

data Annotate a b = Annotate a b

instance Show b => Show (Annotate a b) where
  show (Annotate _ b) = show b

instance Eq b => Eq (Annotate a b) where
  Annotate _ b1 == Annotate _ b2 = b1 == b2

instance Functor (Annotate a) where
  fmap f (Annotate s b) = Annotate s (f b)

instance Applicative (Annotate String) where
  pure x = Annotate "" x
  Annotate s1 f <*> Annotate s2 x = Annotate s2 (f x)
```

Problem 1) The `Annotate` type has two type variables. How are these types used?

Problem 2) Suppose you wanted the `a` type to be shown and considered in equality tests. How would you modify this code to make that happen?

Problem 3) The `Applicative` instance hard-codes a string for the first type. Why is that necessary? (The `FlexibleInstances` extension allows us to do that.)

Rose Trees

Here is a simple tree implementation called a Rose tree.

```haskell
data Rose a = Rose a [Rose a]
            | Empty
```

Problem 4) Implement `Show`

Problem 5) Implement `Eq`

Problem 6) Implement `Functor`

Problem 7) Implement `Applicative`

Type Classes--- Reflector's Report

Manager	Keeps team on track	
Recorder	Records decisions	
Reporter	Reports to Class	
Reflector	Assesses team performance	

1. What was a strength of your team's performance for this activity?

2. What could you do next time to increase your team's performance?

3. What insights did you have about the activity or your team's interaction today?