CS 421 --- State Monad Activity

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

The State Monad

```haskell
data State s a = State { runState :: s -> (a,s) }

instance Monad (State s) where
   return = pure -- or ... return a = State (\s -> (a,s))
   x >>= f = State (\s -> let (y,s2) = runState x s
                              (z,s3) = runState (f y) s2
                          in (z,s3))

get :: State s s
get = State (\s -> (s,s))

put :: a -> State a ()
put x = State (\s -> ((),x))

newState a = State (\s -> (a,s))
```

Problem 1) Notice how when we call pure, we return a `State` function that does not use its state at all. Why is that the right thing to do?

Problem 2) What does the syntax `runState x s` mean?

Problem 3) What is the type of the expression `(f y)`? Why does it have to be that type?

Problem 4) We call `runState` a second time on `(f y)`. We use `s2` in this case. What would happen if we used `s` instead?

Problem 5) Explain what `get` and `put` are doing. Make sure everyone on the team understands them.

Using the State Monad

Here are the `Functor` and `Applicative` definitions for `State`, for reference.

```
instance Functor (State s) where
  fmap f x = State (\s -> let (y,s2) = runState x s
                          in (f y, s2))

instance Applicative (State s) where
  pure a = State (\s -> (a,s))
  ff <*> xx = State (\s -> let (f,s2) = runState ff s
                               (x,s3) = runState xx s2
                           in (f x, s3))
```

Problem 6) Write a function `cplus :: Num a => State s a -> State s a -> State s a` that takes two state integers and adds them, also incrementing the state.

```
Prelude> Main.runState (cplus (newState 10) (newState 20)) 0
(30,1)
```

Problem 7) `get` and `put` are boring. Write `push :: a -> State [a] ()` and `pop :: State [s] s`. You can use `get` and `put` in your definition if you want. Here is a sample function that uses it.

```
addStack x = do
   a <- x
   b <- pop
   push (a + b)
   return b

Prelude> Main.runState (addStack (newState 10)) [5,6]
(5,[15,6])
```

State Monad Activity --- Team's Assessment

Manager or Reflector: Consider the objectives of this activity and your team's experience with it, and then answer the following questions after consulting with your team.

1. What was a strength of this activity? List one aspect that helped it achieve its purpose.

2. What change could we make to this activity to make it more effective?

3. What insights did you have about the activity at the meta level? (I.e., we're not asking abou the content, but maybe how the activity was organized)

State Monad Activity--- 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?