The State Monad

```haskell
0 data State s a = State { runState :: s -> (a,s) }
1 instance Monad (State s) where
2   return = pure -- or ... return a = State (\s -> (a,s))
3   x >>= f = State (\s -> let (y,s2) = runState x s
4     (z,s3) = runState (f y) s2
5       in (z,s3))
6
7 get :: State s s
8 get = State (\s -> (s,s))
9
10 put :: a -> State a ()
11 put x = State (\s -> ((),x))
12
13 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.

```haskell
instance Functor (State s) where
  fmap f x = State \(s -> let \(y,s2) = \text{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) = \text{runState} ff s \in (f x, s3))
```

**Problem 6** Write a function \(cplus :: \text{Num} a \Rightarrow \text{State} s a \rightarrow \text{State} s a \rightarrow \text{State} s a\) that takes two state integers and adds them, also incrementing the state.

```haskell
Prelude> \text{Main}.\text{runState} (cplus (newState 10) (newState 20)) 0
(30,1)
```

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

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

```haskell
Prelude> \text{Main}.\text{runState} (addStack (newState 10)) [5,6]
(5,[15,6])
```
State Monad Activity--- Reflector’s Report

<table>
<thead>
<tr>
<th>Role</th>
<th>Responsibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manager</td>
<td>Keeps team on track</td>
</tr>
<tr>
<td>Recorder</td>
<td>Records decisions</td>
</tr>
<tr>
<td>Reporter</td>
<td>Reports to Class</td>
</tr>
<tr>
<td>Reflector</td>
<td>Assesses team performance</td>
</tr>
</tbody>
</table>

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?

State Monad Activity --- Team's Assessment (SII)

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 is one things we could do to **improve** this activity to make it more effective?

3. What **insights** did you have about the activity, either the content or at the meta level?