TUTORIAL 6

Question 3 in the exam paper focuses in on a discussion of types.

MyBox

These questions refer to a Scala case class MyBox, defined as follows:

```
case class MyBox[T](value:T) {
     def map[R](f:T=>R) = {
       MyBox(f(value))
     }
}
```

An instance of MyBox simply wraps a value of type T – it doesn't do anything very useful with it. However, we can use it to understand some of the behaviour of Scala library classes such as Option.

1. In what ways is this class similar to Scala's Option type? In what ways is it different? What does the symbol R represent?

2. An instance is created using:

   ```
   val m1 = MyBox(9)
   ```

   What would be output by:

   ```
   println m1
   ```

Note - a case class allows you to do the following:
Create an instance without using the *new* key word, as in this example
Return a string representation of itself, in the form MyBox(*value*), when the instance name is written as an expression

3. What would be the result of calling the *map* method of MyBox as follows? Don't guess – look at the definition of *map* in the class and think about what T,R and *f* will be in this example.

   ```
   m1.map{ x => x*x }
   ```

   Similarly, what would be the result of the following call?

   ```
   m1.map{ x => x%3 == 0 }
   ```

4. What would be the result of the following call? Note that the function applied here wraps the result of the calculation in a MyBox.

   ```
   m1.map(x => MyBox(x*x))
   ```

5. The following method is added to MyBox:

```
def flatMap[R](f:T=>MyBox[R]) = {
    map(f).value
}
```

What would be the result of the following call? Again, don't guess – look carefully at the definition of *flatMap*.

```
m1.flatMap(x => MyBox(x*x))
```

6. The class MyBox now has the behaviour of a functor and a monad, similar to Scala library classes such as Option and List. Which method provides the functor behaviour? Which one provides the monad behaviour? Explain why each of functors and monads are useful.

7. The above examples use functions to calculate the square of the value in a MyBox and to check whether the value is divisible by 3. The following expression tries to compose these functions to check whether the square of the value is divisible by 3. Why will it not work? How could this be modified to work, without changing the functions

```
m1.map(x => MyBox(x*x)).map( y => y%3 == 0)
```

8. The following expression combines the values in two MyBox objects:

```
m1.flatMap { x =>
  m2.map { y => x * y }
}
```

If *m2* is defined as:

```
val m2 = MyBox(4)
```

the result is:

```
MyBox(36)
```

9. Write an equivalent for comprehension that produces the same result for *m1* and *m2*.