1. Creating a traffic light class (50 points): Consider the following class `RYGTrafficLight`:

```java
// The " RYG " prefix stands for the Red, Yellow, and Green colors
// that this light can assume.
public class RYGTrafficLight {
    // Colors are encoded as integers
    private final int RED = 1;
    private final int YELLOW = 2;
    private final int GREEN = 3;

    // The current color of the traffic light
    private int color ;

    // The constructor ensures that the light has a valid color
    // ( starts out red )
    public RYGTrafficLight () {
        color = RED ;
    }

    //...add change() method here...

    // Render the individual lamps
    public String drawLamps () {
        if ( color == RED ) {
            return "(R) ( ) ( ) ";
        } else if ( color == GREEN ) {
            return " ( ) ( ) (G)";
        } else if ( color == YELLOW ) {
            return " ( ) (Y) ()";
        } else {
            return " ** Error ** "; // Defensive programming
        }
    }
}
```
Your assignment is to change and add to this class definition in the following ways:

- Add another state, OFF. (Note, we computer science folks like to use 0 to indicate off.)
- Without changing the functionality of color, reduce the space needed to store the state (Hint: consider how many different states we need to be able to store)
- Add a one-argument constructor that allows the user to create a new RYGTrafficLight object that is either RED or OFF (and the variable passed to this constructor matches neither of these states, it should be have the same as the no-argument constructor.)
- To use the constants RED, GREEN, etc, in the constructor above, you must also change those constants to class variables instead of instance variables(as they are given the initial code.)
- Add a method change() that changes the state of the traffic light object. The change() will change the color of the traffic light from its current color to the color that should follow its current color in the normal order of a traffic light’s cycle. (Assume the light follows the pattern of traffic lights in the US.) You must also decide how your change() method should operate if the state of the light is OFF. (Be sure to add a comment that fully describes the behavior of the method you created!!)
- Change the drawLamps() method so that it creates a multi-line, vertical ASCII graphic representing the state of the traffic light (instead of the current horizontal, single-line version). (Be sure to update it to handle the OFF state!) As we don’t want to expose this method to outside users, you will also change the visibility of this class to private.

2. Creating a rational number class (50 points): A rational number is defined as the ratio of two integers, where the second integer must be non-zero. Commonly referred to as a fraction, a rational number’s two integer components are called numerator and denominator. Rational numbers may be added to, subtracted from, multiplied by, or divided by another rational number.

Your assignment is to create a class RationalNumber that:

- Has two private data fields to store the numerator and the denominator
• Provides a two-argument constructor that takes in two integers, the first should be used as the value of the numerator and the second as the value of the denominator. This constructor should check that the second integer is a valid, denominator; if not, the constructor should set the denominator to 1.

• Provides getters for the numerator and denominator.

• Provides a `toString` method that takes no arguments and returns a String with a crude ASCII representation of the rational number.

• Provides an `add` method that takes a variable of type `RationalNumber` returns a variable of type `RationalNumber` that is the sum of itself and the parameter variable. (Note that this method should not change the `RationalNumber` itself nor should it change the parameter variable. I.e. you will need to create a new `RationalNumber` object inside this method.)

• Provides a `subtract` method that takes a variable of type `RationalNumber` returns a variable of type `RationalNumber` that is the difference, itself minus the rational number in the the parameter variable. This `subtract` method should use the `add` method to perform the majority of its job. (Again, this method should not change the `RationalNumber` itself nor should it change the parameter variable.)

• Provides a `multiply` method that takes a variable of type `RationalNumber` returns a variable of type `RationalNumber` that is the product of itself and the parameter variable. (Again, this method should not change the `RationalNumber` itself nor should it change the parameter variable.)

• Provides a `divide` method that takes a variable of type `RationalNumber` returns a variable of type `RationalNumber` that is the quotient, itself divided by the rational number in the the parameter variable. This `divide` method should use the `multiply` method to perform the majority of its job. (Again, this method should not change the `RationalNumber` itself nor should it change the parameter variable.)

Note that you DO NOT need to provide a method that reduces the rational number to lowest terms. You will be adding this feature to your `RationalNumber` class in Homework 6 (after we’ve learned about recursion!).