1. (2 pts) Is the following a valid method declaration? Why or why not?

   public gcd(int firstNumber, int secondNumber) {
   
   No, it is not a valid declaration. You have to have a return type when you declare a method.

2. (2 pts) Is there a compile-time error in the following method declaration? If so, what is the problem?

   public boolean largerThan50(int number) {
   if (number < 50)
       return(false);
   else if (number > 50)
       return(true);
   
   Yes, there is a compile-time error. When there are multiple paths through a method, we have
   to make sure that there is a proper return statement in each path. In the method we don’t
   specify what should happen if number is equal to 50.

3. (2 pts) Does the following code produce a compile-time error? If not, could you add one line
   of code that would cause a compile-time error? Explain. This code is an example of
   methodoverloading which involves having more than one method in a class definition with the
   same name.

   public class Question3 {
       public static int largest(int i, double d) {
           return(1);
       }
       public static int largest(double d, int i) {
           return(1);
       }
       public static void main(String[] args) {
       }
   }

   There is not a compile-time error in this code. The two methods have different signatures.
   However, if we add the call largest(1,1) in the main method, there will be a compile-time error
   because the call will match both methods but neither is more specific than the other.

4. (2 pts) The signature of a method is a combination of its name and parameter list, and all
   parameter passing in Java is by value.
5. (2 pts) What does the following code print? Explain the output.

```java
public class Question5 {
    public static void modify(int number) {
        number = 1;
    }
    public static void main(String[] args) {
        int number = 2;
        modify(number);
        System.out.println("The modified value of number is "+number+".");
    }
}
```

The code prints "The modified value of number is 2." Because all parameter passing in Java is by value, there is no way to affect the value of a primitive like int by sending it as the parameter to a method.