# ISIT 324 Module 4 Homework

**30 Points Possible**

In this homework assignment you’ll explore TDD principles with the moderately famous triangle problem. Here’s how it will work. I can’t look over your shoulder, but if you do it this way you’ll get the most out of it.

Here’s what you want to develop: a method that, given the lengths of three sides of a triangle, can tell whether the triangle is equilateral, isosceles, scalene, or is not a triangle at all. You will build a Triangle class with three properties (SideA, SideB and SideC) and one method, the Analyze method, which returns a string.

So: once Triangle is initialized with the lengths of its three sides, you can run the Analyze method. Its rules are pretty straightforward:

* The length of each side must be a positive number. If any are not positive numbers, throw an application exception.\*
* No side can have a length that is >= the sum of the other two sides. If this condition isn’t true, the three lengths cannot represent a triangle (the edges won’t be able to close.) When this happens, throw an application exception.\*
* If all three sides are of equal length, return the string value “Equilateral.”
* If only two sides are of equal length, return the string value “Isosceles.”
* If all three sides are of different lengths, return the string value “Scalene.”

Here’s how you should proceed:

1. Build your VS solution with a Class project for SUT and a test project for the tests.
2. In the class project, build the Triangle class create the method header for the Analyze method that will evaluate the relationship between a triangle’s three sides. At this point, Analyze should do nothing but return some artificial value. *This sets you up for the* ***Red*** *phase.* Place a comment above this method that reads “*This version of the analyze method supports the Red phase*.”
3. In the test project, build tests to ensure that the program properly identifies whether the three side lengths form a triangle and, if they do, whether the triangle is equilateral, isosceles or scalene. *Remember to use our test naming convention so it’s really, really easy to tell what the test is supposed to be doing!*
4. Run the tests. They should all fail. ***Take a screen shot of your test explorer.***
5. Comment out the Analyze stub (I want to be able to see the commented out code it in your .cs file.)
6. Now write a new version of the Analyze method. This time, make it do what it’s supposed to do. *That sets us up for the* ***Green*** *phase.*
7. Run your tests again. They should all pass. It they don’t, it means that your code doesn’t yet match the specification (at least, theoretically).

**Submit your answer as:**

1. a .zip file of your VS Solution, and
2. a screen shot of your Red phase test pass from the test explorer.

**For full credit:**

* There must be at least one test for each condition specified as a rule, above.
* You must include a screen shot of the Red phase test pass.
* Your SUT file must include both the finished Analyze method *and* the commented-out stub version prefixed with the comment above in 2, above.
* When I unzip the solution you submit, the tests must run. Make sure you zip up all the right stuff!
* Use the recommended name formulation for each of the test methods.
* Arrange your test methods using the “arrange, act, assert” pattern (including comments).
* Use the SUT alias to identify the software under test.

\*If you’d like, you can use a user-defined exception (or exceptions) rather than the generic “ApplicationException” class. If you do, it’ll be worth a couple of points of extra credit.