Assignment Description: In this assignment you will start with an existing implementation of the classify triangle program that will be given to you.   You will also be given a starter test program that tests the classify triangle program, but those tests are not complete.

Author: Brandon Seidman

Summary:

* Summary: I tested at each step to find bugs that were in each part of the testing for the different triangles. I used a method of coming up with tests that would check different edge cases to ensure I didn’t miss anything. After testing each part, I found bugs for all of the parts of the triangle, except for isosceles, and also found some bugs located throughout the code such as testing for negatives. After running 9 tests, one for each then a test to see if fixed for each except isosceles, I caught all the bugs
* Reflection: I learned a lot about making sure my tests covered every case. There were a few bugs that if I didn’t go that extra mile and put in some weird edge cases I would not have caught. For example, I had the most test cases for scalene but only caught one bug. This also taught me a lot about how to test. I started not going in order with the code but realized this was inefficient and fixed my testing practices.

Honor Pledge: I pledge my Honor that I have abided by the Stevens Honor System. -BS

Detailed Results:

* Techniques: Writing tests, seeing if they worked, reporting on the results, updating the code, and making sure it works.
* Assumptions/Constraints: I assumed that it would take in only numbers for the tests but did not assume that it couldn’t be equations. I set this as the restraint for my test cases otherwise I would have needed many more.
* Data Inputs: I used 3 integer numbers for all my tests but used one equation input for each test.
* Explanation of Results:

A close up of text on a white background

Description automatically generated

Above is the results of all the tests when first ran. You can follow when I ran the tests by the test numbers (There are also tests in the next screenshot). You can see here where I ran into errors and when I found these errors, I went into the code to fix them and ran the tests again with the results in the next screenshot.

A close up of text on a white background

Description automatically generated  
This screenshot shows all the tests after I went into the code and fixed the errors. The only anomaly here is Test 9 which was just copied from the original code tests because there were no errors.

A screenshot of a cell phone

Description automatically generated

This matrix is an easier way to follow all the tests. You can see for each problem how many problems were actually causing it and when they were fixed. You can also keep up with how many tests I did overall in each step.