Assignment Description:
Sometimes, you are given a program that someone else has written and you will be asked to fix, update and enhance the program. In this assignment, we had an existing implementation of the classify triangle program, as well as a starter test program. We had to determine if the program was correctly implemented. In order to do this, we had to update the set of test cases in the test program so that the tests adequately tested all of the conditions and once this was done the test cases had to be run against the given implementation of classify triangle. Based on the results of this, then update the classify triangle program to fix all defects and continue to run the test cases until all the defects are fixed.

Author:
Brittany DiFede

Summary:

Matrix:

	Test Run 1	Test Run 2	Test Run 3	Test Run 4	Test Run 5	Test Run 6
Tests Planned	27 (24 of my own + 3 given)	27 (24 of my own + 3 given)	27 (24 of my own + 3 given)	27 (24 of my own + 3 given)	27 (24 of my own + 3 given)	27 (24 of my own + 3 given)
Tests Executed	27	27	27	27	27	27
Tests Passed	6	6	17	21	25	27
Defects Found	11	1	1	6	2	0
Defects Fixed	11	1	1	6	2	0

For this assignment, I first made test cases in order to test the original given implementation of the code. Once I completed that, I was able to use these test cases in order to see where I needed to change the original implementation. Before I made changes, out of 27 tests the code only passed 6. I found 11 different defects and fixed these. However, after I tested the fixed code, I ran the tests again and still only passed 6 out of 27. I looked through the code and saw that I had to fix the requirements where b >= 0 since in the code it stated that b = b. After fixing this, in test run 3 I passed 17 out of 27 tests. Here, I was able to again narrow down what was going wrong and saw that I was spelling Isosceles incorrectly, which was causing certain test cases to fail. After fixing this, I ran test run 4 and was passing 21 out of 27 test cases. I noticed that I was failing all the cases that included right triangles. I was able to

change my code in order for the isosceles and scalene to see if these were right triangles as well, since equilateral triangles can't be right triangles. When I ran this, my test run 5 passed 25 out of 27 test cases. Here I noticed that the two cases I was failing were for the original test cases given. These test cases were incorrect so I changed these cases. Once I did that, for test run 6 I passed all 27 test cases.

This assignment taught me that unit testing does help to identify issues and see what test cases aren't passing. This helps to isolate issues which made it easier to figure out what exactly within the implementation was incorrect. What I noticed worked was using the test cases to isolate the issues. Isolating issues made it so much easier to see what was going wrong. What I noticed didn't work was that I originally thought I would be able to fix all the problems at once, but this was not the case. However, the unit test helped to catch what I did not originally see and helped me to keep making additional changes.

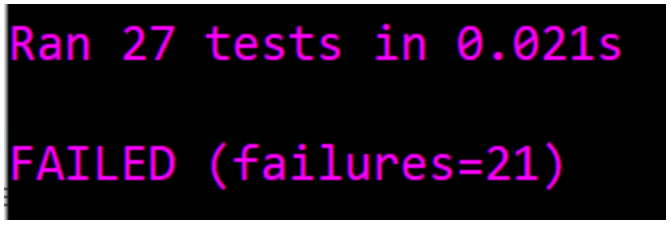
Honor Pledge:
I pledge my honor that I have abided by the Stevens Honor System -Brittany DiFede

Detailed Results:

Test Report for Running my Tests against the Initial Buggy Implementation:

Test ID	Input	Expected Results	Actual Result	Pass or Fail
test_equilateral1	5, 5, 5	Equilateral	InvalidInput	Fail
test_equilateral2	7, 7, 7	Equilateral	InvalidInput	Fail
test_equilateral3	2, 2, 2	Equilateral	InvalidInput	Fail
test_isosceles1	3, 3, 5	Isosceles	InvalidInput	Fail
test_isosceles2	4, 4, 7	Isosceles	InvalidInput	Fail
test_isosceles3	2, 2, 3	Isosceles	InvalidInput	Fail
test_isosceles4	7, 4, 4	Isosceles	InvalidInput	Fail
test_scalene1	4, 5, 6	Scalene	InvalidInput	Fail
test_scalene2	6, 7, 8	Scalene	InvalidInput	Fail
test_scalene3	2, 3, 4	Scalene	InvalidInput	Fail
test_scalene4	6, 5, 4	Scalene	InvalidInput	Fail
test_scaleneright1	3, 4, 5	Scalene Right	InvalidInput	Fail
test_scaleneright2	5, 12, 13	Scalene Right	InvalidInput	Fail

test_scaleneright3	9, 12, 15	Scalene Right	InvalidInput	Fail
test_scaleneright4	5, 4, 3	Scalene Right	InvalidInput	Fail
test_NotTriangle1	3, 2, 8	NotATriangle	InvalidInput	Fail
test_NotTriangle2	1, 2, 7	NotATriangle	InvalidInput	Fail
test_NotTriangle3	4, 3, 9	NotATriangle	InvalidInput	Fail
test_invalidInput1	-2, -3, -5	InvalidInput	InvalidInput	Pass
test_invalidInput2	0, 1, 3	InvalidInput	InvalidInput	Pass
test_invalidInput3	201, 300, 400	InvalidInput	InvalidInput	Pass
test_invalidInput4	1.2, 1.2, 1.2	InvalidInput	InvalidInput	Pass
test_invalidInput5	2.1, 2.1, 3.1	InvalidInput	InvalidInput	Pass
test_invalidInput6	4.2, 5.2, 6.4	InvalidInput	InvalidInput	Pass

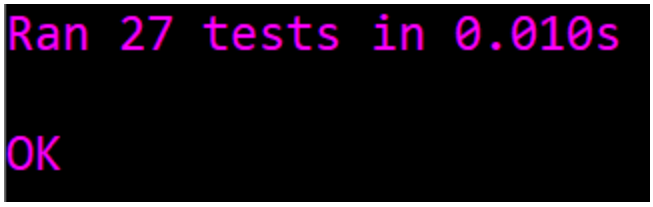


*27 Tests → Include the 3 originally given (Not in table Above)

Test Report for Running my Tests against the improved Implementation:

Test ID	Input	Expected Results	Actual Result	Pass or Fail
test_equilateral1	5, 5, 5	Equilateral	Equilateral	Pass
test_equilateral2	7, 7, 7	Equilateral	Equilateral	Pass
test_equilateral3	2, 2, 2	Equilateral	Equilateral	Pass
test_isosceles1	3, 3, 5	Isosceles	Isosceles	Pass
test_isosceles2	4, 4, 7	Isosceles	Isosceles	Pass
test_isosceles3	2, 2, 3	Isosceles	Isosceles	Pass
test_isosceles4	7, 4, 4	Isosceles	Isosceles	Pass
test_scalene1	4, 5, 6	Scalene	Scalene	Pass

test_scalene2	6, 7, 8	Scalene	Scalene	Pass
test_scalene3	2, 3, 4	Scalene	Scalene	Pass
test_scalene4	6, 5, 4	Scalene	Scalene	Pass
test_scaleneright1	3, 4, 5	Scalene Right	Scalene Right	Pass
test_scaleneright2	5, 12, 13	Scalene Right	Scalene Right	Pass
test_scaleneright3	9, 12, 15	Scalene Right	Scalene Right	Pass
test_scaleneright4	5, 4, 3	Scalene Right	Scalene Right	Pass
test_NotTriangle1	3, 2, 8	NotATriangle	NotATriangle	Pass
test_NotTriangle2	1, 2, 7	NotATriangle	NotATriangle	Pass
test_NotTriangle3	4, 3, 9	NotATriangle	NotATriangle	Pass
test_invalidInput1	-2, -3, -5	InvalidInput	InvalidInput	Pass
test_invalidInput2	0, 1, 3	InvalidInput	InvalidInput	Pass
test_invalidInput3	201, 300, 400	InvalidInput	InvalidInput	Pass
test_invalidInput4	1.2, 1.2, 1.2	InvalidInput	InvalidInput	Pass
test_invalidInput5	2.1, 2.1, 3.1	InvalidInput	InvalidInput	Pass
test_invalidInput6	4.2, 5.2, 6.4	InvalidInput	InvalidInput	Pass



*Including the 3 given test cases (two of those modified & none in the table above)

In regards to the results, when I first added my own test cases none of them passed when running them against the original implementation of the code. Every single test case outputted InvalidInput no matter what the actual implementation was supposed to be.

In order to fix the code, I used the technique of looking at the test cases that didn't pass and going back into the code where these cases wouldn't pass in order to identify the issues that the code was causing. This technique worked for me, as after each test run I was able to isolate the issues in the code even further until I was able to fix the code completely and pass every test case.

In regards to the constraints, I looked at the constraints already in place within the code and checked to see if the logic behind these constraints was accurate. While doing this, I noticed some of the logic to be incorrect. When doing this, I also noticed the issue with how the right triangle was set up. For this, it had in the code that a triangle had to be right or scalene or isosceles or equilateral when in reality a scalene triangle and an isosceles triangle can also be a right triangle. Due to this, I fixed the constraint put in place. I also additionally fixed the logic for other constraints, but besides the right constraint didn't notice many other constraints to be missing.

For the data inputs for my test cases, I tried to implement anything that I believed a user could possibly input. This included inputs that were correct as well as incorrect. By doing this, I wanted to ensure that my code worked as it should and picked up any errors that could potentially cause the code to break.

These results proved to me that unit testing does work. As seen from my results, the first time I ran my test cases majority failed whereas the last time I ran my test cases none failed. This helped to show me how successful and beneficial unit testing is in identifying issues and producing correct code.