1. **Assignment Description**:

Sometimes you will be given a program that someone else has written, and you will be asked to fix, update and enhance that program. 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.

In order to determine if the program is correctly implemented, you will need to update the set of test cases in the test program. You will need to update the test program until you feel that your tests adequately test all of the conditions. Then you should run the complete set of tests against the original triangle program to see how correct the triangle program is. Capture and then report on those results in a formal test report described below. For this first part you should not make any changes to the classify triangle program. You should only change the test program.

Based on the results of your initial tests, you will then update the classify triangle program to fix all defects. Continue to run the test cases as you fix defects until all of the defects have been fixed. Run one final execution of the test program and capture and then report on those results in a formal test report described below.

2. **Author**: Anshul Kapoor

GitHub Repo: https://github.com/anshulkapoor018/Triangle567

3. **Summary**

Initial test results:

Test ID	Input	Expected Results	Actual Results	Pass or Fail
testEquilateralTriangle1	*(2, 2, 2)*	Equilateral	InvalidInput	Fail
testEquilateralTriangle2	*(15, 15, 15)*	Equilateral	InvalidInput	Fail
testEquilateralTriangle3	*(10, 1, 10)*	Equilateral	**Equilateral**	Pass
testIsoscelesTriangle1	*(5, 5, 3)*	Isosceles	InvalidInput	Fail
testIsoscelesTriangle2	*(4, 6, 6)*	Isosceles	InvalidInput	Fail
testIsoscelesTriangle3	*(8, 5, 8)*	Isosceles	InvalidInput	Fail
testIsoscelesTriangle4	*(6, 6, 4)*	Isosceles	InvalidInput	Fail
testScaleneTriangle1	*(10, 11, 12)*	Scalene	InvalidInput	Fail
testScaleneTriangle2	*(4, 2, 3)*	Scalene	InvalidInput	Fail
testScaleneTriangle3	*(100, 110, 112)*	Scalene	InvalidInput	Fail
testScaleneTriangle4	*(10, 10, 12)*	Scalene	**Scalene**	Pass
testInvalidInput1	*(-1, -1, -1)*	InvalidInput	**InvalidInput**	Pass
testInvalidInput3	*("200", "0", "0")*	InvalidInput	InvalidInput	Fail
testNotATriangle1	*(5, 1, 1)*	NotATriangle	InvalidInput	Fail
testNotATriangle2	*(1, 5, 1)*	NotATriangle	InvalidInput	Fail
testNotATriangle3	*(1, 1, 5)*	NotATriangle	InvalidInput	Fail
testNotATriangle4	*(1, 17, 5)*	NotATriangle	InvalidInput	Fail
testRightTriangle1	*(3, 4, 5)*	RightTriangle	InvalidInput	Fail
testRightTriangle2	*(5, 3, 4)*	RightTriangle	InvalidInput	Fail
testRightTriangle3	*(13, 12, 5)*	RightTriangle	InvalidInput	Fail
testRightTriangle4	*(8, 6, 10)*	RightTriangle	InvalidInput	Fail
testRightTriangle5	*(21, 6, 10)*	RightTriangle	**RightTriangle**	Pass

Test Run Matrix:

	Test Run 1	Test Run 2	Test Run 3	Test Run 4
Tests Planned	22	22	22	22
Tests Executed	22	22	22	22
Tests Passed	4	6	18	22
Defects Found	2	1	3	0
Defects Fixed	0	2	1	3

Final test results:

Test ID	Input	Expected Results	Actual Results	Pass or Fail
testEquilateralTriangle1	*(2, 2, 2)*	Equilateral	Equilateral	Pass
testEquilateralTriangle2	*(15, 15, 15)*	Equilateral	Equilateral	Pass
testEquilateralTriangle3	*(10, 1, 10)*	Equilateral	Equilateral	Pass
testIsoscelesTriangle1	*(5, 5, 3)*	Isosceles	Isosceles	Pass
testIsoscelesTriangle2	*(4, 6, 6)*	Isosceles	Isosceles	Pass
testIsoscelesTriangle3	*(8, 5, 8)*	Isosceles	Isosceles	Pass
testIsoscelesTriangle4	*(6, 6, 4)*	Isosceles	Isosceles	Pass
testScaleneTriangle1	*(10, 11, 12)*	Scalene	Scalene	Pass
testScaleneTriangle2	*(4, 2, 3)*	Scalene	Scalene	Pass
testScaleneTriangle3	*(100, 110, 112)*	Scalene	Scalene	Pass
testScaleneTriangle4	*(10, 10, 12)*	Scalene	Scalene	Pass
testInvalidInput1	*(-1, -1, -1)*	InvalidInput	InvalidInput	Pass
testInvalidInput3	*("200", "0", "0")*	InvalidInput	InvalidInput	Pass
testNotATriangle1	*(5, 1, 1)*	NotATriangle	NotATriangle	Pass
testNotATriangle2	*(1, 5, 1)*	NotATriangle	NotATriangle	Pass
testNotATriangle3	*(1, 1, 5)*	NotATriangle	NotATriangle	Pass
testNotATriangle4	*(1, 17, 5)*	NotATriangle	NotATriangle	Pass
testRightTriangle1	*(3, 4, 5)*	RightTriangle	RightTriangle	Pass
testRightTriangle2	*(5, 3, 4)*	RightTriangle	RightTriangle	Pass
testRightTriangle3	*(13, 12, 5)*	RightTriangle	RightTriangle	Pass
testRightTriangle4	*(8, 6, 10)*	RightTriangle	RightTriangle	Pass
testRightTriangle5	*(21, 6, 10)*	RightTriangle	RightTriangle	Pass

Test-driven debugging is a very effective way of fixing erroneous code. As I fixed defects in the code and ran the tests, more defects became apparent. However, writing tests as you write code would be a more effective way of error-checking than writing all the code, then all the tests in my opinion.

4. Honor pledge:

I pledge my honor that I have abided by the Stevens Honor System.