1. **Assignment Description**:

You could occasionally be asked to enhance, modernize, or fix software that was created by someone else. A pre-existing copy of the categorize triangle program will be provided to you at the start of the project. You will also get a beginner test program that only fully evaluates the classify triangle program.

To determine whether the software is appropriately implemented, you must modify the test program's collection of test cases. You must update the test program until you feel that all of the conditions have been adequately tested by your testing. Then you should run every test against the original triangle program to see how accurate it is. Record the findings in the official test report, which is outlined below, after keeping track of the outcomes. For this initial step, you shouldn't make any changes to the existing triangle software. It is sufficient to simply modify the test program..

Based on the results of your initial testing, you will then update the categorize triangle program to fix every error. As you make corrections, continue running the test cases until all problems have been fixed. Run the test program one last time, note the results, and then include a report on them in the formal test report that is specified below.

2. **Author**: Shashank Ramesh Kumar

GitHub: https://github.com/RK-ops/Triangle567

3. Summary

Initial test results:

Test ID	Input	Expected Results	Actual Results	Pass or Fail
testEquilateralTriangles	*(1, 1, 1)*	Equilateral	InvalidInput	Fail
testEquilateralTriangle1	*(7, 7, 7)*	Equilateral	InvalidInput	Fail
testEquilateralTriangle2	*(15, 1, 15)*	Equilateral	Equilateral	Pass
testIsoscelesTriangle1	*(5, 5, 3)*	Isosceles	InvalidInput	Fail
testIsoscelesTriangle2	*(4, 6, 6)*	Isosceles	InvalidInput	Fail
testIsoscelesTriangle3	*(8, 6, 8)*	Isosceles	InvalidInput	Fail
testIsoscelesTriangle4	*(6, 6, 4)*	Isosceles	InvalidInput	Fail
testScaleneTriangle1	*(10, 11, 12)*	Scalene	InvalidInput	Fail
testScaleneTriangle2	*(1, 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
testInvalidInput2	*("200", "200", "200")*	InvalidInput	InvalidInput	Fail
testNotATriangle1	*(1, 3, 5)*	NotATriangle	InvalidInput	Fail
testNotATriangle2	*(1, 4, 5)*	NotATriangle	InvalidInput	Fail
testNotATriangle3	*(1, 0, 1)*	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	*(1, 1, 1)*	Equilateral	Equilateral	Pass
testEquilateralTriangle2	*(7, 7, 7)*	Equilateral	Equilateral	Pass
testEquilateralTriangle3	*(15, 1, 15)*	Equilateral	Equilateral	Pass
testIsoscelesTriangle1	*(5, 5, 3)*	Isosceles	Isosceles	Pass
testIsoscelesTriangle2	*(4, 6, 6)*	Isosceles	Isosceles	Pass
testIsoscelesTriangle3	*(8, 6, 8)*	Isosceles	Isosceles	Pass
testIsoscelesTriangle4	*(6, 6, 4)*	Isosceles	Isosceles	Pass
testScaleneTriangle1	*(10, 11, 12)*	Scalene	Scalene	Pass
testScaleneTriangle2	*(1, 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
testInvalidInput2	*("200", "200", "200")*	InvalidInput	InvalidInput	Pass
testNotATriangle1	*(1, 3, 5)*	NotATriangle	NotATriangle	Pass
testNotATriangle2	*(1, 4, 5)*	NotATriangle	NotATriangle	Pass
testNotATriangle3	*(1, 0, 1)*	NotATriangle	NotATriangle	Pass
testNotATriangle4	*(1, 17, 5)*	NotATriangle	NotATriangle	Pass
testRightTriangle1	*(3, 4, 5)*	Right	Right	Pass
testRightTriangle2	*(5, 3, 4)*	Right	Right	Pass
testRightTriangle3	*(13, 12, 5)*	Right	Right	Pass
testRightTriangle4	*(8, 6, 10)*	Right	Right	Pass
testRightTriangle5	*(21, 6, 10)*	Right	Right	Pass

Test-driven debugging is a very efficient way to rectify incorrect code. More errors were found as I ran the tests and fixed issues in the code. However, I think that the order of writing tests and writing all the code is a more effective way to error-check than the other way around.

4. **Honor pledge:**

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