1. **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.

2. **Author**: Yulong Yan

3. **Summary**:

Buggy report:

Test ID	Input	Expected Results	Actual Result	Pass or Fail
testRightTriangleA	(3,4,5)	'Right'	'InvalidInput'	Fail
testRightTriangleB	(5,3,4)	'Right'	'InvalidInput'	Fail
testEquilateralTriangles	(1,1,1)	'Equilateral'	'InvalidInput'	Fail
testIsocelesTriangle	(6,6,4)	'Isoceles'	'InvalidInput'	Fail
testScaleneTriangle	(2,8,9)	'Scalene'	'InvalidInput'	Fail
testNotTriangle1	(6,6,14)	'NotATriangle'	'InvalidInput'	Fail
testNotTriangle2	(2,4,2)	'NotATriangle'	'InvalidInput'	Fail
testInvalid1	(2.1,3.1,4.1)	'InvalidInput'	'InvalidInput'	Pass
testInvalid2	(-3,-4,-5)	'InvalidInput'	'InvalidInput'	Pass
testInvalid3	(300,500,600)	'InvalidInput'	'InvalidInput'	Pass
testInvalid4	(0,2,3)	'InvalidInput'	'InvalidInput'	Pass
testInvalid5	(0,0,0)	'InvalidInput'	'InvalidInput'	Pass

Correct report:

Test ID	Input	Expected Results	Actual Result	Pass or Fail
testRightTriangleA	(3,4,5)	'Right'	'Right'	Pass
testRightTriangleB	(5,3,4)	'Right'	'Right'	Pass
testEquilateralTriangles	(1,1,1)	'Equilateral'	'Equilateral'	Pass
testIsocelesTriangle	(6,6,4)	'Isoceles'	'Isoceles'	Pass
testScaleneTriangle	(2,8,9)	'Scalene'	'Scalene'	Pass
testNotTriangle1	(6,6,14)	'NotATriangle'	'NotATriangle'	Pass
testNotTriangle2	(2,4,2)	'NotATriangle'	'NotATriangle'	Pass
testInvalid1	(2.1,3.1,4.1)	'InvalidInput'	'InvalidInput'	Pass
testInvalid2	(-3,-4,-5)	'InvalidInput'	'InvalidInput'	Pass
testInvalid3	(300,500,600)	'InvalidInput'	'InvalidInput'	Pass
testInvalid4	(0,2,3)	'InvalidInput'	'InvalidInput'	Pass
testInvalid5	(0,0,0)	'InvalidInput'	'InvalidInput'	Pass

a summary of your results:

	Test Run 1	Test Run 2
Tests Planned	(3,4,5),(5,3,4),(1,1,1),(6,6,4) (2,8,9),(6,6,14),(2,4,2),(2.1,3.1,4.1) (-3,-4,-5),(300,500,600),(0,2,3) (0,0,0)	(3,4,5),(5,3,4),(1,1,1),(6,6,4) (2,8,9),(6,6,14),(2,4,2),(2.1,3.1,4.1) (-3,-4,-5),(300,500,600),(0,2,3) (0,0,0)
Tests Executed	(3,4,5),(5,3,4),(1,1,1),(6,6,4) (2,8,9),(6,6,14),(2,4,2),(2.1,3.1,4.1) (-3,-4,-5),(300,500,600),(0,2,3) (0,0,0)	(3,4,5),(5,3,4),(1,1,1),(6,6,4) (2,8,9),(6,6,14),(2,4,2),(2.1,3.1,4.1) (-3,-4,-5),(300,500,600),(0,2,3) (0,0,0)
Tests Passed	(3,4,5),(5,3,4),(1,1,1),(6,6,4) (2,8,9),(6,6,14),(2,4,2)	(3,4,5),(5,3,4),(1,1,1),(6,6,4) (2,8,9),(6,6,14),(2,4,2),(2.1,3.1,4.1) (-3,-4,-5),(300,500,600),(0,2,3) (0,0,0)

Defects Found		if a <= 0 or b <= b or c <= 0:
		if (a >= (b - c)) or (b >= (a - c)) or (c >= (a + b)):
		if a == b and b == a:
		elif ((a * 2) + (b * 2)) == (c * 2):

Defects Fixed		if a <= 0 or b <= 0 or c <= 0:
		if c >= a + b or a >= b + c or b >= a + c:
		if a == b and b == c:
		elif a**2 + b**2 == c**2 or a**2 + c**2 == b**2 or b**2 + c**2 == a ** 2:

- reflection :

I did learn how to find bugs using the test cases(unit tests)

5. **Honor pledge**

6. **Detailed results, if any:**