Testing a Legacy Program and Reporting on Testing Results

Assignment Description:

The objective of this assignment is to develop a set of test for an existing triangle classification program.Of these tests, the objective is to find and fix defects in the program. Afterward, report our results from the triangle problem.

Author: Michael Buglione

Summary:

After looking at the code for some time I was able to divide up the testing into two sections. Test 1 covers the data that is being input it into the triangle as such that is allowable for the code. An example of this is if the variables a b and c which are designated as the sides of the triangle are in between 1 and 200 and are as well able to be formed into a triangle with their designated lengths.To figure out the errors in the code two functions were made testtriangle which tested if the triangle had valid input and testvalidtriangle which had tested if the triangle was valid in lengths. Test 2 covers the pairing of what type of triangle would be the result of three valid length, ab&c. I was able to cover test twos subjects by using tests testRightTriangleA, testRightTriangle and testEquilateralTriangles.From these two tests I was able to find all the errors that were produced in the initial code.

Test Run 1

Test ID	Input	Expected Results	Actual Result	Pass or Fail
testRightTriangleA	classifyTriangle(3,4,5)	'Right'	InvalidInput	Fail
testRightTriangle	classifyTriangle(5,4,3)	'Right'	InvalidInput	Fail
testEquilateralTriangles	classifyTriangle(1,1,1)	'Equilateral'	InvalidInput	Fail
testTriangle	classifyTriangle(1,1,1)	'InvalidInput'	Failure	Fail
testvalidTriangle	classifyTriangle(50,3,4)	'NotATriangle'	InvalidInput	Fail

Test Run 2

Test ID	Input	Expected Results	Actual Result	Pass or Fail
testRightTriangleA	classifyTriangle(3,4,5)	'Right'	'Right'	Fail
testRightTriangleB	classifyTriangle(5,4,3)	'Right'	'Right'	Fail
testEquilateralTriangles	classifyTriangle(1,1,1)	'Equilateral'	'Equilateral'	Fail
testTriangle	classifyTriangle(1,1,1)	'InvalidInput'	'InvalidInput'	Pass
testvalidTriangle	classifyTriangle(50,3,4)	'NotATriangle'	'NotATriangle'	Pass

Matrix

	Test Run 1	Test Run 2
Tests Planned	To test if the program allows for other values than ints that are greater than 0 and if the values could be an triangle	To test if the program choose the proper trangles for the project.
Tests Executed	testvalidTriangle testTriangle	testRightTriangleA testRightTriangle testEquilateralTriangles
Tests Passed	none	None

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):
		Added :
		a, b, c = sorted([a, b, c]) #sorts the values in the triangle

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

Reflection

Reflection of its overall assignment is that I learned how to use UnitTest and testing in a more realistic sense. I didn't have any real problems while working on this assignment. Although due to the Simplicity of the assignment it was a little hard to have a more in-depth description on the testing process. I believe in this assignment what works best for me was reading over the code and breaking up into parts that could be tested separately such as having the units to make the triangle have its own test and then having the assigning of triangles in different tests. Overall I thought this was a great project and a great intro to unit testing in a more realistic environment.

I pledge my honor that I provided by the Stevens honor System

Michael Buglione