Erik Buczek

SSW-567 SW Testing Qual. Assur. & Maintenance

02/05/19

HW 02a - Testing a legacy program and reporting on testing results

**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.

**Summary:**

**Test 1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Input | Expected Result | Actual Result | Pass or Fail |
| RightTriangleA | (3,4,5) | Right | InvalidInput | Fail |
| RightTriangleB | (5,3,4) | Right | InvalidInput | Fail |
| EquilateralTriangles | (1,1,1) | Equilateral | InvalidInput | Fail |
| ScaleneTriangle | (2,3,4) | Scalene | InvalidInput | Fail |
| IsocelesTriangle | (1,1,2) | Isoceles | Right | Fail |
| ValidA | (201,1,2) | InvalidInput | InvalidInput | Pass |
| ValidB | (1,201,2) | InvalidInput | InvalidInput | Pass |
| ValidC | (1,2,201) | InvalidInput | InvalidInput | Pass |
| ValidD | (0,1,2) | InvalidInput | InvalidInput | Pass |
| ValidE | (1,0,2) | InvalidInput | InvalidInput | Pass |
| ValidF | (1,2,0) | InvalidInput | InvalidInput | Pass |
| ValidG | (1.5,1,2) | InvalidInput | InvalidInput | Pass |
| ValidH | (1,1.5,2) | InvalidInput | InvalidInput | Pass |
| ValidI | (1,2,1.5) | InvalidInput | InvalidInput | Pass |
| ValidJ | (3,1,2) | NotATriangle | InvalidInput | Fail |
| ValidK | (1,3,2) | NotATriangle | InvalidInput | Fail |
| ValidL | (1,2,3) | NotATriangle | InvalidInput | Fail |

**Test 2**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Input | Expected Result | Actual Result | Pass or Fail |
| RightTriangleA | (3,4,5) | Right | Right | Pass |
| RightTriangleB | (5,3,4) | Right | Right | Pass |
| RightTriangleC | (4,5,3) | Right | Right | Pass |
| EquilateralTriangleA | (1,1,1) | Equilateral | Equilateral | Pass |
| EquilateralTriangleB | (2,2,2) | Equilateral | Equilateral | Pass |
| ScaleneTriangleA | (2,3,4) | Scalene | Scalene | Pass |
| ScaleneTriangleB | (12,15,23) | Scalene | Scalene | Pass |
| IsocelesTriangleA | (2,2,3) | Isoceles | Isoceles | Pass |
| IsocelesTriangleB | (67,67,100) | Isoceles | Isoceles | Pass |
| ValidA | (201,1,2) | InvalidInput | InvalidInput | Pass |
| ValidB | (1,201,2) | InvalidInput | InvalidInput | Pass |
| ValidC | (1,2,201) | InvalidInput | InvalidInput | Pass |
| ValidD | (0,1,2) | InvalidInput | InvalidInput | Pass |
| ValidE | (1,0,2) | InvalidInput | InvalidInput | Pass |
| ValidF | (1,2,0) | InvalidInput | InvalidInput | Pass |
| ValidG | (1.5,1,2) | InvalidInput | InvalidInput | Pass |
| ValidH | (1,1.5,2) | InvalidInput | InvalidInput | Pass |
| ValidI | (1,2,1.5) | InvalidInput | InvalidInput | Pass |
| ValidJ | (3,1,2) | InvalidInput | InvalidInput | Pass |
| ValidK | (1,3,2) | InvalidInput | InvalidInput | Pass |
| ValidL | (1,2,3) | InvalidInput | InvalidInput | Pass |

**Results Matrix**

|  |  |  |
| --- | --- | --- |
|  | Test Run 1 | Test Run 2 |
| Tests Planned | 17 | 21 |
| Tests Executed | 17 | 21 |
| Tests Passed | 9 | 21 |
| Defects Found | 8 | 0 |
| Defects Fixed | 8 | 0 |

I decided that the 21 tests I wrote were sufficient based on the number of cases defined by the classifyTriangle function. I split the types of tests into two categories: valid triangles and invalid triangles. There were four different if statements that defined whether an input would be “InvalidInput” or “NotATriangle”. I found that these statements needed 12 tests altogether in order to sufficiently test each case. The valid triangle set needed 9 different tests. I kept the original tests “valid” that were provided, and I added 6 more of my own.

**Reflection:**

I found this assignment to be much more straightforward as compared to the first assignment. At this point, I am already comfortable with the basics of Python and the tools associated with it, and so I did not need to scramble to learn any new information as I did previously. The original Triangle.py file had a mostly functioning code, from my perspective. I only needed to change a few things in order to fully run my tests. In terms of the test cases, I found it very simple to just add more test cases until I felt satisfied that all of the cases were accounted for. Additionally, I found the usage of Github for this assignment to be very straightforward and easy, as was the case for the last assignment.

**Detailed Results:**

The results for this assignment are very simple, they are outlined in the test tables provided above. In terms of the techniques I used, I made use of the Pycharm IDE for both editing and testing purposes. The function and its conditional statements only needed to be changed once in order to clear up all existing bugs. The data inputs for this assignment were the side lengths of the triangle, however these inputs were subject to the test cases.

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