Project Summary

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.

* These are the two files:  Triangle.py and TestTriangle.py
  + [***Triangle.py***](https://sit.instructure.com/courses/30792/files/4506927/download?verifier=jLYmOGOtUMBSHsZaB8s0Uv6K2OEd4E8Kot4R5PC4&wrap=1)is a starter implementation of the triangle classification program.
  + [***TestTriangle.py***](https://sit.instructure.com/courses/30792/files/4506926/download?verifier=I3T0FJXx8f5B5iNZZTOHN9KSJHYfJ4wPAQTG7IEP&wrap=1)**c**ontains a starter set of unittest test cases to test the classifyTriangle() function in the file Triangle.py file.

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.

Note that you should NOT simply replace the logic with your logic from Assignment 1. Test teams typically don't have the luxury of rewriting code from scratch and instead must fix what's delivered to the test team.

[*Triangle.py*](https://sit.instructure.com/courses/30792/files/4506927/download?verifier=jLYmOGOtUMBSHsZaB8s0Uv6K2OEd4E8Kot4R5PC4&wrap=1)*contains an implementation of the classifyTriangle() function with a few bugs.*  
  
[*TestTriangle.py*](https://sit.instructure.com/courses/30792/files/4506926/download?verifier=I3T0FJXx8f5B5iNZZTOHN9KSJHYfJ4wPAQTG7IEP&wrap=1)*contains the initial set of test cases*

1. Author: Ziang Lin
2. Summary:

I executed 2 runs of testing, added several new test cases which it wasn’t covered previously and found out 6 defects in the classifyTriangle() function. After testing and bug fixing, the function can be run properly.

|  |  |  |
| --- | --- | --- |
|  | Test Run 1 | Test Run 2 |
| Tests Planned | Run the test and check if it covered all the situation | Check if the bug fixed after first run and continuous fixing the bugs of shape identification |
| Tests Executed | 9 test cases and only passed 1 case. Found out 6 defects in the function, included valid check, shape identification, formula | All 9 tests cases passed and fixed 6 bugs in the function and improved the performance of valid check feature. |
| Test Passed | 1 | 9 |
| Defects Found | 6 | 0 |
| Defects Fixed | 3 | Fix the bugs found in first run |

1. Refection

I practiced how to use UnitTest to locate bugs in the function. When considering testcases included, it’s easier to list all the possibility in case missing any testcase. After 2 runs of testing, the function can be run properly and totally found 6 defects in the original function. Add 6 more testcases in the test script.

1. Honor pledge

I pledge on my honor that I have not given or received any unauthorized assistance on this assignment/examination. I further pledge that I have not copied any material from a book, article, the Internet or any other source except where I have expressly cited the source.

Original test report

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Input | Expected Results | Actual Result | Pass or Fail |
| 1 | 1,1,2 | NotATriangle | InvalidInput | Fail |
| 2 | 3,4,5 | Right | InvalidInput | Fail |
| 3 | 5,3,4 | Right | InvalidInput | Fail |
| 4 | 1,1,1 | Equilateral | InvalidInput | Fail |
| 5 | 6,7,8 | Scalene | InvalidInput | Fail |
| 6 | -1,1,1 | InvalidInput | InvalidInput | Pass |
| 7 | a, b, c | InvalidInput | TypeError | Fail |
| 8 | 2,2,sqrt(8) | Right and Isosceles | InvalidInput | Fail |
| 9 | 2,5,sqrt(29) | Right and Scalene | InvalidInput | Fail |

Improved test report

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Input | Expected Results | Actual Result | Pass or Fail |
| 1 | 1,1,2 | NotATriangle | NotATriangle | Pass |
| 2 | 3,4,5 | Right | Right | Pass |
| 3 | 5,3,4 | Right | Right | Pass |
| 4 | 1,1,1 | Equilateral | Equilateral | Pass |
| 5 | 6,7,8 | Scalene | Scalene | Pass |
| 6 | -1,1,1 | InvalidInput | InvalidInput | Pass |
| 7 | a,b,c | InvalidInput | InvalidInput | Pass |
| 8 | 2,2,sqrt(8) | Right and Isosceles | Right and Isosceles | Pass |
| 9 | 2,5,sqrt(29) | Right and Scalene | Right and Scalene | Pass |