Homework 02a

In this assignment, I develop a set of tests for an existing triangle classification program, use those tests to find and fix defects in that program, and report on my testing results for the Triangle problem.

By Benjamin Knobloch

**Contained in repo**: <https://github.com/beknobloch/LegacyProgramTestingExemplar>

My augmented test suite revealed that the base implementation of ClassifyTriangle was badly error-ridden, failing over half of the test cases it was subject to. After repairing several unique bugs in the algorithm, my updated version of ClassifyTriangle passes all twenty test cases and can correctly classify triangles.

In this assignment, I learned the value of subjecting code to pre-drafted test cases rather than relying on live testing. A prime example of this is the spelling mistake in “isosceles” in the original version of the function, which I may have missed if I were live testing in the console but which my automated test functions flagged as a problem (“Isoseles” != “Isosceles”). Additionally, I gained practice with the iterative process of bug development in which one suite of test cases is repeatedly used and bugs are tracked, documented, and repaired between each run. This led to a smoother, more recognizable sense of progress while bug fixing than can be achieved with informal or live testing. One issue with the scheme of this assignment was the imprecise feedback from invalid inputs. A more rigorous program might throw errors with a clear indication of why the input was invalid rather than simply returning “Invalid input” or “Not a triangle,” which would improve the program from both the testing and user-facing angles.

**Test set against original classifyTriangle implementation:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Input | Expected Results | Actual Result | Pass or Fail |
| 1 | 3, 4, 5 | Right | Invalid input | Fail |
| 2 | 5, 3, 4 | Right | Invalid input | Fail |
| 3 | 1, 1, 1 | Equilateral | Invalid input | Fail |
| 4 | -1, -1, -1 | Invalid input | Invalid input | Pass |
| 5 | 0, 0, 0 | Invalid input | Invalid input | Pass |
| 6 | 3, 4, 0 | Invalid input | Invalid input | Pass |
| 7 | 3, 4, -1 | Invalid input | Invalid input | Pass |
| 8 | 0, 3, 4 | Invalid input | Invalid input | Pass |
| 9 | -1, 3, 4 | Invalid input | Invalid input | Pass |
| 10 | 3, 0, 4 | Invalid input | Invalid input | Pass |
| 11 | 3, -1, 4 | Invalid input | Invalid input | Pass |
| 12 | 10, 10, 10 | Equilateral | Invalid input | Fail |
| 13 | 5, 5, 3 | Isosceles | Invalid input | Fail |
| 14 | 10, 11, 12 | Scalene | Invalid input | Fail |
| 15 | 10, 10, 150 | Not a triangle | Invalid input | Fail |
| 16 | 5, 5, 5 | Equilateral | Invalid input | Fail |
| 17 | 10, 10, 15 | Isosceles | Invalid input | Fail |
| 18 | 10, 12, 14 | Scalene | Invalid input | Fail |
| 19 | 5, 12, 13 | Right | Invalid input | Fail |
| 20 | 1, 1, 199 | Not a triangle | Invalid input | Fail |

**Test set against debugged classifyTriangle implementation:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Input | Expected Results | Actual Result | Pass or Fail |
| 1 | 3, 4, 5 | Right | Right | Pass |
| 2 | 5, 3, 4 | Right | Right | Pass |
| 3 | 1, 1, 1 | Equilateral | Equilateral | Pass |
| 4 | -1, -1, -1 | Invalid input | Invalid input | Pass |
| 5 | 0, 0, 0 | Invalid input | Invalid input | Pass |
| 6 | 3, 4, 0 | Invalid input | Invalid input | Pass |
| 7 | 3, 4, -1 | Invalid input | Invalid input | Pass |
| 8 | 0, 3, 4 | Invalid input | Invalid input | Pass |
| 9 | -1, 3, 4 | Invalid input | Invalid input | Pass |
| 10 | 3, 0, 4 | Invalid input | Invalid input | Pass |
| 11 | 3, -1, 4 | Invalid input | Invalid input | Pass |
| 12 | 10, 10, 10 | Equilateral | Equilateral | Pass |
| 13 | 5, 5, 3 | Isosceles | Isosceles | Pass |
| 14 | 10, 11, 12 | Scalene | Scalene | Pass |
| 15 | 10, 10, 150 | Not a triangle | Not a triangle | Pass |
| 16 | 5, 5, 5 | Equilateral | Equilateral | Pass |
| 17 | 10, 10, 15 | Isosceles | Isosceles | Pass |
| 18 | 10, 12, 14 | Scalene | Scalene | Pass |
| 19 | 5, 12, 13 | Right | Right | Pass |
| 20 | 1, 1, 199 | Not a triangle | Not a triangle | Pass |

**Test run scheme:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Test Run 1 | Test Run 2 | Test Run 3 |
| Tests Planned | 20 | 20 | 20 |
| Tests Executed | 20 | 20 | 20 |
| Tests Passed | 8 | 17 | 20 |
| Defects Found | 4 | 2 | 0 |
| Defects Fixed | 4 | 2 | 0 |

I decided I had enough test cases once every conditional endpoint had been tested at least twice, and when inputs had to be validated, when every input was validated. For example, I tested passing in zero and negative values for each of the three parameters of the function, tested each type of triangle output with two different sets of values, and used the same sets of numbers in different orders to validate input symmetry.

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

**Assumptions**

* If a triangle is a right triangle, it should be classified as “right” alone and not further classified as “right scalene” or “right isosceles.”
* When positive side lengths are given, but they cannot constitute a valid triangle, the result should be “Not a triangle” rather than “Invalid input.”

**Result Notes**

* I counted six unique defects that I repaired in the original implementation of ClassifyTriangle.
* The defects included mistaken variable references (e.g. making the comparison b <= b rather than b <= 0), not treating inputs symmetrically (e.g. checking for a right triangle with c as the hypotenuse, but not as a or b as the hypotenuse) or typos (a misspelling of “isosceles”).