Mikayla Mount
SSW 567 Quality Assurance
Dr. Bondi
9/19/2022
HW02a

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

- **Assignment Description:**
  - Fix, update and enhance an existing triangle classification program. A starter test program that tests the classification triangle program is also provided, but those tests are not complete. Test the logic and results of both of these programs, document the results, then fix the classification program and write complete test cases. Document these results and the progress made during this process, both in a report and on GitHub.
- **Summary**
  - This assignment highlighted several vital themes with working on buggy, legacy code. The first most obvious takeaway is the process of working with coding styles that differ from yours, which requires the user to employ more time and attention to both the style and the flow of semantic logic in their code. Apart from the specific style, this demonstrated how essential comments are to convey the logic and intent when writing code that others will see, especially for this task. Using these breadcrumbs and persistent trial and error, I was able to find a solution that worked for testing a variety of triangle classifications.
- **Detailed Results**
  - Based on my prior knowledge of triangles, I knew that the format in which the tests were written were correct and that my work lay in the Triangle.py file. The initial test results, while still failures, provided direction to where I should start, since it was made clear that both the equilateral and right triangle tests were incorrect. In addition to the resultant hints, the comments also helped immensely understand what the intent of the function was, and helped to show me where the initial developer made mistakes. This legacy code that was given was indeed riddled with logical errors, as seen in Table 1, where all tests failed. In my first several attempts to fix them I struggled, as seen in the first three test runs in Table 3. As seen in the later tests in Table 3, I was able to get the legacy code after a very iterative testing process. In short, most every functional line needed to be tweaked in one way or another, and only through purposeful trial and error was I able to find a solution that would appease the test code.

Table 1:
PART 1

Test ID	Input	Expected Result	Actual Result	Pass or Fail
Right_A	(3,4,5)	Right	InvalidInput	Fail
Right_B	(5,3,4)	Right	InvalidInput	Fail
Equil_A	(1,1,1)	Equilateral	InvalidInput	Fail

Table 2:
PART 2

Test ID	Input	Expected Result	Actual Result	Pass or Fail
Right_A	(3,4,5)	Right	Right	Pass
Right_B	(5,3,4)	Right	Right	Pass
Equil_A	(1,1,1)	Equilateral	Equilateral	Pass
Equil_B	(6,6,6)	Equilateral	Equilateral	Pass
Scalene_A	(7,8,10)	Scalene	Scalene	Pass
Scalene_B	(4,5,6)	Scalene	Scalene	Pass
Isosceles_A	(5,5,7)	Isosceles	Isosceles	Pass
Isosceles_B	(6,6,8)	Isosceles	Isosceles	Pass
NotATri_A	(1,2,3)	NotATriangle	NotATriangle	Pass
NotATri_B	(1,5,6)	NotATriangle	NotATriangle	Pass
Invalid_A	(0,2,3)	InvalidInput	InvalidInput	Pass
Invalid_B	(2.5,5 ,6)	InvalidInput	InvalidInput	Pass
Invalid_C	(-9,5,6)	InvalidInput	InvalidInput	Pass

Table 3:
PROGRESS MATRIX

	Test Run 1	Test Run 2	Test Run 3	Test Run 4	Test Run 5
Tests Planned	3	3	5	6	10
Tests Executed	3	3	5	6	10
Tests Passed	0	0	3	6	10
Defects Found	3	3	2	0	0
Defects Fixed	0	0	3	0	0