Triangle\_Classifier Application: Test Plan

Test Plan draft ChatGPT LLM generated then manually updated

Python Application and Test code draft Copilot LLM generated then manually updated

**1. Introduction**

The **Triangle\_Classifier** application reads input strings and classifies them as one of the following types of triangles: "Scalene", "Isosceles", "Equilateral", or "NoTriangle". The application also handles error cases and exits on specific input commands ("Exit" or "Quit"). This test plan outlines the steps required to qualify the application for delivery, including all test suites and cases, pass/fail criteria, and test data.

**Application Specifics**

**Inputs**: New line separated strings. Words in a string are separated by white space.

WHEN there are three separate words in a string, AND when each word is a number, then the application outputs one word.

**Outputs**: one word from the set of words "Scalene", "Isosceles", "Equilateral", or "NoTriangle” on stdout. **Error** strings on stderr.

**Operation**: The choice of the output word is made by interpreting each number as a length of a side of a triangle.

In **ALL OTHER** cases the output is an error string indicating that the input could not be classified.

The error string also indicates the cause of the error.

Causes include what it is about the string that it cannot be classified OR execution errors.

The program exits with a stdout done message when the input string contains the word “**Exit**” or the word “**Quit**” (ignore case) at the beginning of the string.

Examples:

Input “1 1 1” Output “Equilateral”

Input “1” Output “Error: Two sides missing”

Input “Quit” Application exits.

**1.1 Objective**

The objective of this test plan is to ensure that the **Triangle\_Classifier** application:

* Correctly classifies valid triangle inputs.
* Accurately reports errors for invalid input strings.
* Properly exits on receiving "Exit" or "Quit".
* Functions as expected under various input scenarios.

**1.2 Scope**

The tests will cover:

- Functional validation for correct triangle classification.

- Error handling for invalid input strings.

- Proper application termination when given "Exit" or "Quit".

**2. Test Suites and Test Cases**

**2.1 Test Suite 1: Valid Triangle Classification**

**Test Case 1.1: Equilateral Triangle Integer**

**Description**: Verify that the application correctly identifies equilateral triangles where all three sides are equal.

* **Input**: `1 1 1`
* **Expected Output**: `Equilateral`
* **Pass Criteria**: Application outputs "**Equilateral**".

**Test Case 1.2: Isosceles Triangle Integer**

**Description** Verify that the application correctly identifies isosceles triangles where two sides are equal.

* **Input**: `2 2 3`
* **Expected Output** `Isosceles`
* **Pass Criteria**: Application outputs "Isosceles".

**Test Case 1.3: Scalene Triangle Integer**

**Description**: Verify that the application correctly identifies scalene triangles where all three sides are different.

* **Input**: `5 3 4`
* **Expected Output**: `Scalene`
* **Pass Criteria**: Application outputs "Scalene".

**Test Case 1.4: NoTriangle Integer**

**Description**: Verify that the application correctly identifies cases where the three sides do not form a triangle.

* **Input**: `2 3 1`
* **Expected Output**: `NoTriangle`
* **Pass Criteria**: Application outputs "NoTriangle".

**Test Case 1.5: Equilateral Triangle Float**

**Description**: Verify that the application correctly identifies equilateral triangles where all three sides are equal with tab white space

* **Input**: `12.3 12.3 12.3`
* **Expected Output**: `Equilateral`
* **Pass Criteria**: Application outputs "**Equilateral**".

**Test Case 1.6: Isosceles Triangle Float**

**Description** Verify that the application correctly identifies isosceles triangles where two sides are equal.

* **Input**: `45.6 23.4 23.4
* **Expected Output** `Isosceles`
* **Pass Criteria**: Application outputs "Isosceles".

**Test Case 1.7: Scalene Triangle Float**

**Description**: Verify that the application correctly identifies scalene triangles where all three sides are different with tab white space

* **Input**: `3.6 5.9 4.8`
* **Expected Output**: `Scalene`
* **Pass Criteria**: Application outputs "Scalene".

**Test Case 1.8: NoTriangle Float**

**Description**: Verify that the application correctly identifies cases where the three sides do not form a triangle

* **Input**: `7.8 1.1 2.3 `
* **Expected Output**: `NoTriangle`
* **Pass Criteria**: Application outputs "NoTriangle".

**Test Case 1.9: NoTriangle Zero**

**Description**: Verify that the application correctly identifies cases where at least one of three sides is zero, so does not form a triangle with tab white space

* **Input**: `1.1 2 0.0`
* **Expected Output**: `NoTriangle`
* **Pass Criteria**: Application outputs "NoTriangle".

**Test Case 1.10: NoTriangle Negative**

**Description**: Verify that the application correctly identifies cases where at-least one of three sides is negative, so does not form a triangle with tab white space

* **Input**: `7.7 -0.13 7.8`
* **Expected Output**: `NoTriangle`
* **Pass Criteria**: Application outputs "NoTriangle".

**2.2 Test Suite 2: Error Handling**

**Test Case 2.1: Missing Two Sides only**

**Description**: Verify that the application returns an error when fewer than three numbers are provided.

* Input: `1.1`
* **Expected Output**: `Error: Two sides missing`
* Pass Criteria: Application outputs the correct error message.

**Test Case 2.2: Missing One Side only**

**Description**: Verify that the application returns an error when only two numbers are provided.

* Input: `5.6 10`
* Expected Output: `Error: One side missing`
* Pass Criteria: Application outputs the correct error message.

**Test Case 2.3: Missing Three Sides only**

**Description**: Verify that the application returns an error when no numbers are provided.

* Input: ` `
* **Expected Output**: `Error: Three sides missing`
* Pass Criteria: Application outputs the correct error message.

**Test Case 2.4: More than Three Sides numbers**

**Description**: Verify that the application returns an error when more than three numbers are provided.

* Input: `1 2 3.1 5.6`
* **Expected Output**: `Error: More than Three words provided`
* Pass Criteria: Application outputs the correct error message.

**Test Case 2.5: Invalid three non-numbers**

**Description**: Verify that the application returns an error when only non-numeric input is provided.

* Input: `all three words`
* Expected Output: `Error: Non-numeric words`
* Pass Criteria: Application outputs the correct error message.

**Test Case 2.6: Invalid two non-numbers**

**Description**: Verify that the application returns an error when only non-numeric input is provided.

* Input: `only 2two`
* Expected Output: `Error: Non-numeric words`
* Pass Criteria: Application outputs the correct error message.

**Test Case 2.7: More than three non-numbers**

**Description**: Verify that the application returns an error when only non-numeric input is provided.

* Input: More 1than three3 inputs`
* Expected Output: `Error: More than Three words provided `
* Pass Criteria: Application outputs the correct error message.

**Test Case 2.8: Missing One Side with non- number**

**Description**: Verify that the application returns an error when only two numbers are provided.

* Input: `5.6 number 10`
* Expected Output: `Error: One side missing with Non-numeric words`
* Pass Criteria: Application outputs the correct error message.

**Test Case 2.9: Missing Two Sides with 1 non- number**

**Description**: Verify that the application returns an error when fewer than three numbers are provided.

* Input: `non 1.1`
* **Expected Output**: `Error: Two sides missing with Non-numeric words`
* Pass Criteria: Application outputs the correct error message.

**Test Case 2.10: Missing Two Sides with 2 non- numbers**

**Description**: Verify that the application returns an error when fewer than three numbers are provided.

* Input: `non 1.1 number`
* **Expected Output**: `Error: Two sides missing with Non-numeric words`
* Pass Criteria: Application outputs the correct error message.

**2.3 Test Suite 3: Application Termination**

**Test Case 3.1: Quit Command**

**Description**: Verify that the application exits when the input string only has "Quit".

* Input: `Quit`
* Expected Output: triangle classifier done.
* Pass Criteria: Application terminates successfully with output message

**Test Case 3.2: Quit second word**

**Description**: Verify that the application has Error when the input string has "Quit" as second input.

* Input: `1 Quit 3`
* Expected Output: Error: One side missing with Non-numeric words
* Pass Criteria: Application terminates successfully.

**Test Case 3.3: Quit first lowercase**

**Description**: Verify that the application exits when the input string on has "quit" as first word

* Input: `quit 2 3 `
* Expected Output: triangle classifier done.
* Pass Criteria: Application terminates successfully with output message

**Test Case 3.4: Quit starts first word**

**Description**: Verify that the application has Error when the input string has "Quit" as second input.

* Input: `Quit1 4 3.5`
* Expected Output: Error: One side missing with Non-numeric words
* Pass Criteria: Application terminates successfully.

**Test Case 3.5: Exit Command**

**Description**: Verify that the application exits when the input string contains "Exit".

* Input: `Exit`
* Expected Output: triangle classifier done.
* Pass Criteria: Application terminates successfully with output message

**Test Case 3.6: Exit third word**

**Description**: Verify that the application has Error when the input string has "Exit" as third input.

* Input: `1 A Exit`
* Expected Output: Error: Two sides missing with Non-numeric words
* Pass Criteria: Application terminates successfully.

**Test Case 3.7: Exit first uppercase**

**Description**: Verify that the application exits when the input string contains "Exit".

* Input: `EXIT 3. 2`
* Expected Output: triangle classifier done.
* Pass Criteria: Application terminates successfully with output message

**Test Case 3.8: Exit ends first word**

**Description**: Verify that the application has Error when the input string has "Quit" as second input.

* Input: `1Exit 4 3.5`
* Expected Output: Error: One side missing with Non-numeric words
* Pass Criteria: Application terminates successfully.

**Test Case 3.9: Valid Invalid then Quit**

**Description**: Verify that the has Valid and Invalid Triangles then exits

* Input lines
  1. 23 34 45.6
  2. 23.4 78.4 19
  3. 1.0 3.0
  4. Quit
* Expected stdout lines
  1. Scalene
  2. NoTriangle
  3. triangle classifier done.
* Expected stderr lines
  1. Error: Missing One side
* Pass Criteria: Application terminates successfully with output and error messages

**Test Case 3.10: Valid Invalid then Exit**

**Description**: Verify that the has Valid and Invalid Triangles then exits

* Input lines

2. 3.4 5.6 3.40
3. 4.8
4. 5.7 5.70 5.700
5. Exit

* Expected stdout lines
  1. Isosceles
  2. Equilateral
  3. triangle classifier done.
* Expected stderr lines

1. Error: Missing Three sides
2. Error: Missing Two sides

* Pass Criteria: Application terminates successfully with output and error messages

**2.4 Test Suite 4: Boundary and Edge Cases**

**Test Case 4.1: Very Large Inputs**

**Description**: Verify how the application handles large integer inputs.

* Input: `999999999 999999999 999999999`
* Expected Output: `Equilateral`
* Pass Criteria: Application outputs "Equilateral" for large valid inputs.

**Test Case 4.2: Floating Round Scalene**

**Description**: Verify that the application handles float many decimals.

* Input: ` "3.00000000000001 5.000 8.00\n", **note handles 14 decimals**
* **Expected Output**: `Scalene`
* **Pass Criteria**: Application outputs "Scalene".

**Test Case 4.3: Floating Round No Triangle**

**Description**: Verify that the application rounds after many decimals.

* Input: "3.000000000000001 5.000 8.00\n", **note rounds at 15 decimals**
* **Expected Output**: ` NoTriangle`
* **Pass Criteria**: Application outputs " NoTriangle".

**Test Case 4.4: Exit then data**

**Description**: Verify that the application exits and ignores input after exiting".

* Input lines

1. Exit
2. 3.1 5.2 8.01
3. 8.1

* Expected stdout lines

1. triangle classifier done

* Expected stderr line: None
* Pass Criteria: Application exits with done message and no other stdout/stderr data

**3. Test Execution Sequence**

The test cases should be executed in the following sequence:

1. **Test Suite 1**: Validate classification of correct inputs.

2. **Test Suite 2**: Check for various error conditions and input validation.

3. **Test Suite 3**: Test application termination commands.

4. **Test Suite 4**: Test boundary and edge cases.

**4. Criteria for Delivery**

The application will be considered qualified for delivery if:

- All test cases in **Test Suite 1** pass, meaning valid triangles are correctly classified.

- All test cases in **Test Suite 2** pass, meaning errors are accurately detected and reported.

- All test cases in **Test Suite 3** pass, meaning the application exits correctly on receiving "Exit" or "Quit".

- All test cases in **Test Suite 3** pass, meaning the application exits correctly boundary and Edge Cases

**5. Pass/Fail Criteria**

Each test case is deemed to pass if the actual output matches the expected output. For error cases, the output message must correctly indicate the reason for failure, and for valid cases, the classification of the triangle must be correct.

- **Pass**: Application behavior matches the expected behavior.

- **Fail**: Any deviation from the expected behavior, including incorrect classification, missing output, or application crashes.

**6. Data Required to Determine Pass/Fail**

The pass/fail outcome will be determined based on:

- The correct classification output (one of: "Scalene", "Isosceles", "Equilateral", "NoTriangle").

- The error message content and format.

- The ability of the application to properly terminate on "Quit" or "Exit".

All outputs will be manually compared against expected results. Logs (if any) can also be used to trace and verify correct execution paths.

**DONE**