# ISIT 324 Homework 7 – Test Case Design

**45 Points Possible**

In this homework assignment you’ll demonstrate your understanding of specification-based test case design using a variation of the triangle problem from the Module 5 homework. You’ll need to answer a set of questions in this document and submit your answers, and you’ll need to write the updated triangle program and prepare its test cases.

**Problem Definition: Triangle 2**

The goal of the program is the same as the one you wrote for HW 5: detect whether, given three sides of a triangle, the triangle is equilateral, isosceles, scalene, not a triangle at all, or that there is something wrong with the input. The difference is that for this assignment:

* The lengths of the sides must be expressed as integers.
* Valid lengths that can’t product a triangle don’t cause an application exception. Instead, the message “Not a triangle” is produced.
* No side may have a length greater than 200. This is a completely arbitrary restriction that gives you a nice upper bound.

With that in mind, here are the rules: once the triangle is initialized with the lengths of its three sides, you can run the Analyze method. Its rules are similar to those in the previous exercise but note the difference in rules 1 and 2.

1. The length of each side must be a positive number and must be <= 200. If any do not meet these criteria, throw an exception.
2. No side can have a length that is >= the sum of the other two sides. If this condition isn’t true, the three lengths cannot represent a triangle (the edges won’t be able to close.) When this happens, return the string value “Not a triangle.”
3. If all three sides are of equal length, return the string value “Equilateral.”
4. If only two sides are of equal length, return the string value “Isosceles.”
5. If all three sides are of different lengths, return the string value “Scalene.”

**Questions**

1. **(5 points)** Write the values to apply to perform robust boundary value analysis on any one side of the triangle.
2. **(10 points)**. List the normal equivalence classes for the triangle problem in table 2.1 based on the range (output) of the triangle problem. Provide one test case for each equivalence class in table 2.2

*Table 2.1*

|  |
| --- |
| **Equivalence Class** |
| 1: |
| 2: |
| 3: |
| 4: |

*Table 2.2*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case** | **Side a** | **Side b** | **Side c** | **Expected Output** |
| TC1 | … | … | … | … |

1. (**10 points).** Complete the decision table below. It includes a full set of valid Actions for the triangle problem and a couple of example conditions. Your task:
   1. Complete the set of conditions. Do **not** include conditions for invalid values (i.e., lengths < 1 or > 200).
   2. Add the cases that give rise to the actions. (A case is a combination of conditions specified in a column.) **Case 1** is specified for you.

In the **Conditions** section, use T for true, F for false and a hyphen (-) for “don’t care” conditions. Each cell must have a value.

In the **Actions** section, use X for the cells in which the corresponding action (in the row) will take place based on the precipitating conditions (in the column).

Each column has a **Case Number** in its heading. You’ll use that to ensure that each of the decision table cases are accounted for in your unit tests.

Here’s your starting point:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Conditions** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| a <= b + c? | F |  |  |  |  |  |  |  |  |  |  |
| *?* | - |  |  |  |  |  |  |  |  |  |  |
| *?* | - |  |  |  |  |  |  |  |  |  |  |
| a = b? | - |  |  |  |  |  |  |  |  |  |  |
| *?* | - |  |  |  |  |  |  |  |  |  |  |
| *?* | - |  |  |  |  |  |  |  |  |  |  |
| **Actions** |  |  |  |  |  |  |  |  |  |  |  |
| Not a triangle | X |  |  |  |  |  |  |  |  |  |  |
| Scalene |  |  |  |  |  |  |  |  |  |  |  |
| Isosceles |  |  |  |  |  |  |  |  |  |  |  |
| Equilateral |  |  |  |  |  |  |  |  |  |  |  |
| Impossible |  |  |  |  |  |  |  |  |  |  |  |

1. (**10 points)** Write a set of test cases that will allow you to test each of the cases you identified in your decision table and represent them in the table below. Use the decision table’s column label as the Case ID for each row. Each cell in the columns *a, b,* and *c* should be the length of a side. Each cell in column *Expected Output* should correspond to an action from the decision table.

For impossible rules, put an “x” in the values of a, b and c.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Case ID | a | b | c | Expected Ouput |
| 1 | *length of a* | *length of b* | *length of c* | … |
| 2 | … | … | … | … |
| 3 | … | … | … | … |
| etc | … | … | … | … |

1. (**10 points)** Write the triangle program and build a test project that includes the tests required to:
   1. Ensure that the SUT properly handles invalid values for each side.
   2. Ensure that all test cases in your decision table are accounted for.

**For full credit:**

Submission

* Send your answers as a .docx file.
* Send your program as two .cs files:
  + Triangle.cs for the program under test, and
  + TriangleTests.cs for the UnitTest program.
* ***Do not zip your files.***

Requirements

* Use MSTest V2 parameterized tests (i.e., use the DataTestMethod and DataRow attributes.)
* For tests based on the decision table, name each row with a display name that holds the case number from the decision table. Here’s an example from the NextDate exercise:

[DataRow(4, 16, 2001, 4, 15, 2001, DisplayName = "Cases 1 - 3")]

* Standard stuff:
  + Use one test class for the method.
  + Use the recommended name formulation for each of the test methods.
  + Do not use the ExpectedException attribute.
  + Arrange your test methods using the “arrange, act, assert” pattern (including comments).
  + Use the SUT alias to identify the software under test.