TestAnalyze\_Input5\_5\_5\_ExpectEQU

5,5,5 Testing a basic equilateral triangle

TestAnalyze\_Input5\_5\_7\_ExpectISO

5,5,7 Testing a basic isosceles triangle

TestAnalyze\_Input5\_6\_7\_ExpectSCA

5,6,7 Testing a basic scalene triangle

TestAnalyze\_Input\_5\_6\_7\_ExpectNOT

-5,6,7 Testing if there’s a negative value

TestAnalyze\_Input0\_6\_7\_ExpectNOT

0,6,7 Testing if there’s a 0

TestAnalyze\_Input5\_2\_2\_ExpectNOT

5,2,2 Testing if the integers are positive but couldn’t form a triangle

TestAnalyze\_Input6\_5\_7\_ExpectSCA

6,5,7 Testing if a and b’s position would affect the result

TestAnalyze\_Input7\_6\_5\_ExpectSCA

7,6,5 Testing if a and c’s position would affect the result

TestAnalyze\_Input5\_7\_6\_ExpectSCA

5,7,6 Testing if b and c’s position would affect the result

For Program.cs:

M= E− N+ 2P=42-35+2=9

For Analyze:

M= E− N+ 2P=31-25+2=8

This tells me the program.cs is more complex than the analyze. And because 9<10 and 8 <10, the program is easy enough to be written in a single module. For Analyze, 8 is the upper limit for the number of test cases that are necessary to achieve complete branch coverage. (The reason I had 9 test cases though is because I tested the <=0 branch twice, using 0 and a negative integer.)