Summary

A program that determines the type of triangle based of the lengths of the three sides of integers given. It will determine if the triangle is equilateral, isosceles, scalene or a legal triangle. A triangle is a graph that contains three vertices and three edges connected at each vertex.

Oracle Information

Test input will be three integers of possible triangle lengths. Our domain is random numbers from 0 to int.max. The test cases are written in a two dimensional array in our main function where we can change the input values.

This code is written in C language and is tested using an IDE called CodeBlocks.

The output should reflect legal triangle possibilities which includes equilateral, isosceles or scalene.

Oracle Procedure

After the test cases are defined in the two dimensional array, it is run through a for loop which tests each row of three inputs. The program begins by checking if the program is a legal triangle by calculating whether or not the largest side is less than the sum of the other two sides. The sum of the two other sides must be larger than the largest side.

For example: a=2, b=3, and c=5. Since 5 is not less than 2 + 3, these three inputs are invalid to form a legal triangle.

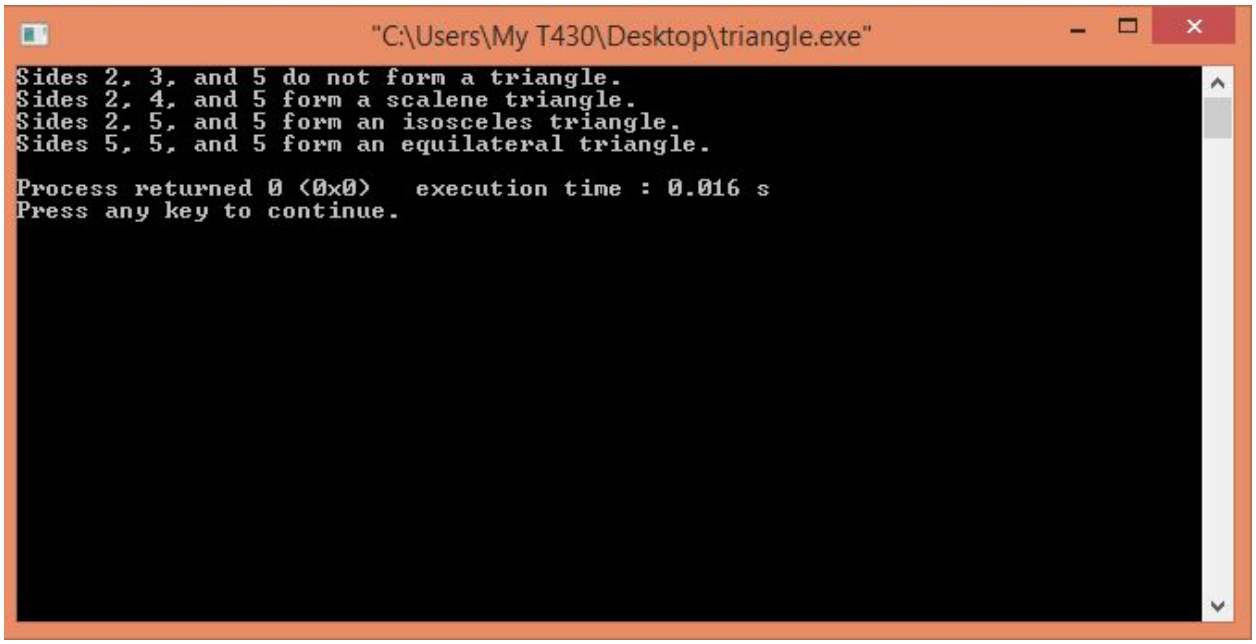
After it finds out if the three integers are a legal triangle, it checks what type of a triangle from the inputs that were given. This is done by using if statements. If the three inputs are equal, the output should be an equilateral triangle. If two of the sides are equal and the third is not, the triangle should be an isosceles, else it's a scalene.

Test Monitor

Here is a sample of the two dimensional array of inputs:

```
int tests[4][3] = {
    {2, 3, 5},
    {2, 4, 5},
    {2, 5, 5},
    {5, 5, 5}
};
```

Here is an example of the output after the program executes with the given values:

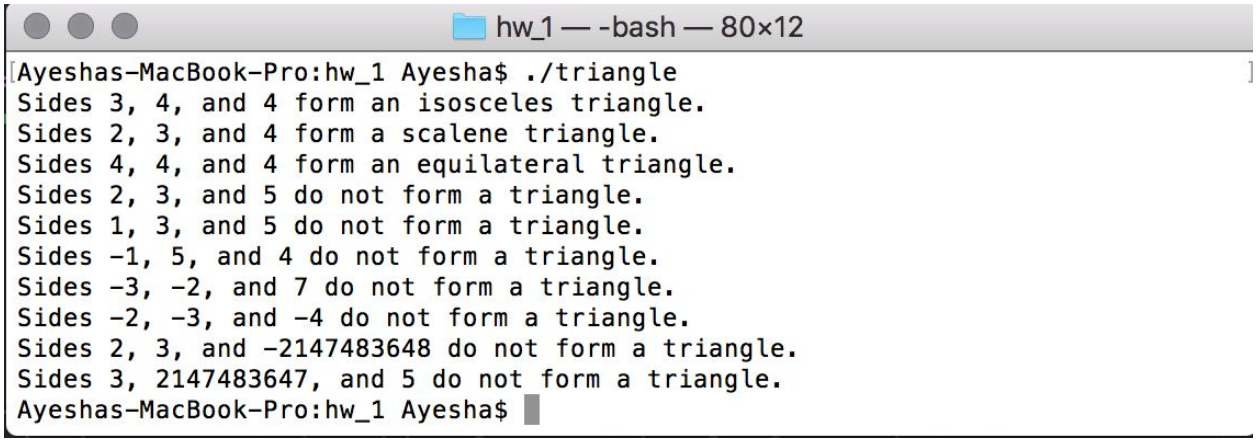


Example Test Cases

Example inputs:

```
58
59          int tests[10][3] = {
60            {3, 4, 4}, //isosceles
61            {2, 3, 4}, //scalene
62            {4, 4, 4}, //equilateral
63            {2, 3, 5}, // fail since 5 = 2 + 3
64            {1, 3, 5}, // fail since 5 > 1 + 3
65            {-1, 5, 4}, //fail since negative
66            {-3, -2, 7}, //fail since negative
67            {-2, -3, -4}, //fail since negative
68            {2, 3, INT_MIN}, //fail
69            {3, INT_MAX, 5}, // fail
70          };
```

Output of the test cases from above:



The first 3 input values were written to show that the program works on the most basic cases. From the 4th case to the 10th case, they prove that the system is able to recognize test cases that should not work.