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

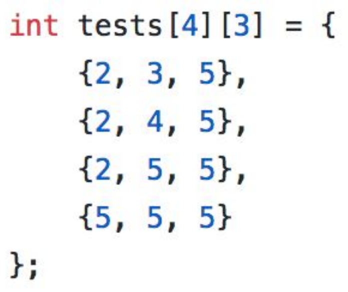
After the test cases are defined in the two dimensional array, it is run through a for loop which tests each row of three inputs. The program begins by checking if the program is a legal triangle by calculating whether or not the largest side is less than the sum of the other two sides. The sum of the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

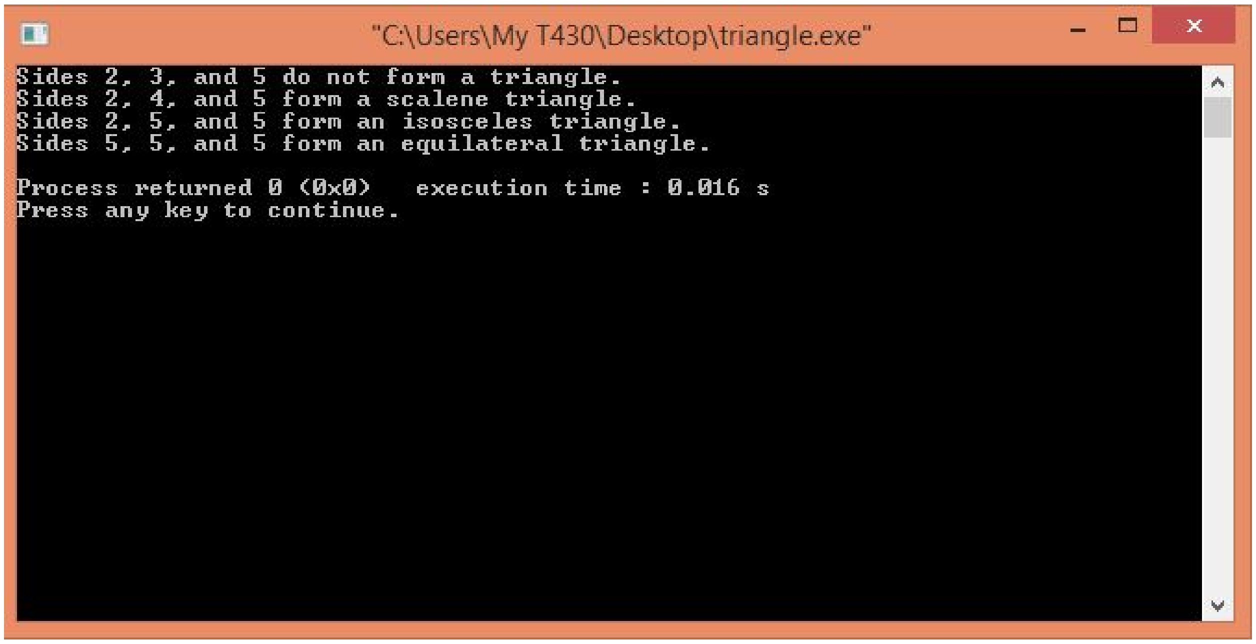
Test Monitor

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



```c
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:



```
Sides 2, 3, and 5 do not form a triangle.
Sides 2, 4, and 5 form a scalene triangle.
Sides 2, 5, and 5 form an isosceles triangle.
Sides 5, 5, and 5 form an equilateral triangle.

Process returned 0 (0x0)   execution time : 0.016 s
Press any key to continue.
```

Test Cases:

Input			**Expected Output**	**Reason for test**
a	**b**	**c**		
3	4	4	isosceles	Ensures that iff two sides are of equal length, then the triangle is recognized as being isosceles
2	3	4	scalene	Ensures that iff no sides are of equal length, then the traingle is recognized as being scalene
4	4	4	equilateral	Ensures that iff all three sides are of equal length, then the triangle is recognized as being equilateral
2	3	5	fail	Ensures that the triangle inequality holds for the case when the sum of the two shorter sides is strictly equal to the longest side
1	3	5	fail	Ensures that the triangle inequality holds for the case when the sum of the two shorter sides is strictly less than the longest side
-1	5	4	fail	Ensures that a case with one negative side cannot be considered a triangle
-3	-2	7	fail	Ensures that a case with two negative sides cannot be considered a triangle
-2	-3	-4	fail	Ensures that a case with three negative sides cannot be considered a triangle

```c
/*
 * Should properly state whether or not 3 sides form a triangle and, if so,
 * what type the triangle is. Please check this for accuracy!
 *
 * Compile: `gcc -std=c99 -o triangles triangles.c`
 * Run: `./triangles`
 */

#include <stdio.h>
#include <string.h>
#include <limits.h>

#define EXPAND_ARGS(args) args[0], args[1], args[2]

int find_largest_side(int a, int b, int c, int * other_sides) {
    /*
     * Finds the largest of three lines, isolates the two remaining lines by
     * copying them into an array, and then returns the largest line.
     */

    int largest = (a > b && a >c ) ? a : (b > c ? b : c);
    memcpy(other_sides,
            (a == largest) ? (int [2]){b, c} :
                    ((b == largest) ? (int [2]){a, c} : (int [2]){a, b}),
            2*sizeof(int));
    return largest;
}

int is_triangle(int a, int b, int c) {
    /*
     * Tests the triangle inequality which states that the largest side of a
     * triangle must be less than the sum of the two remaining sides.
     */

    int others[2];
    if (find_largest_side(a, b, c, others) < others[0] + others[1])
        return 1;
    return 0;
}

char * triangle_type(int a, int b, int c) {
    /*
     * Checks if all sides of the triangle are equal (equilateral), else if any
     * two sides are equal (isosceles), else no sides are equal (scalene).
     * Returns a string stating which type the triangle is.
     */

    return (a == b && b == c) ? "n equilateral" : ((a == b || a == c || b == c)
            ? "n isosceles" : " scalene");
}

int main() {
    /*
     * Defines an array of independent tests where each test contains three
     * integers representing the lengths of the sides of a supposed triangle.
     * Prints whether or not the sides form a triangle for each test.
     */

    int tests[10][3] = {
      {3, 4, 4},        // isosceles
      {2, 3, 4},         // scalene
      {4, 4, 4},         // equilateral
      {2, 3, 5},         // fail since 5 = 2 + 3
      {1, 3, 5},         // fail since 5 > 1 + 3
      {-1, 5, 4},        // fail since negative
      {-3, -2, 7},       // fail since negative
      {-2, -3, -4},      // fail since negative
      {2, 3, INT_MIN},   // fail
      {3, INT_MAX, 5},   // fail
    };

    for (int i=0; i < sizeof(tests)/sizeof(tests[0]); i++)
        if (is_triangle(EXPAND_ARGS(tests[i])))
            printf("Sides %d, %d, and %d form a%s triangle.\n",
                    EXPAND_ARGS(tests[i]),
                    triangle_type(EXPAND_ARGS(tests[i])));
        else
            printf("Sides %d, %d, and %d do not form a triangle.\n",
                    EXPAND_ARGS(tests[i]));
}
```