Mohd. Aftab Alam
02013302717

EXPERIMENT 3

Aim :

(a) To determine the type of Triangle. Its input is triple of positive integers (say a, b, c) and values may be from the interval [1, 100]. The output may have one of the following :

Equilateral, Isosceles, Scalene or Not a Triangle.

Perform Boundary Value Analysis and show the Test Cases.

Algorithm :

- Take 3 inputs from the user for the sides of the Triangle.
- Check whether they lie in the given interval.
- If the condition is false, stop the program and exit.
- If the condition is true, check the type of Triangle.
- If all three sides are equal, Equilateral Triangle.
- If any two sides are equal, Isosceles Triangle.
- If all three sides are different, Scalene Triangle
- According to the formula ***4 n + 1,*** there will be ***13*** test cases, where n is number of inputs.

Code :

```
#include <iostream>

using namespace std;

void bva(int, int, int);

int main()
{
    int min, max;
    int x, y, z;
    cout << "Enter Range : ";
    cin >> min >> max;

    if (min < 0 || max > 100)
    {
        cout << "Invalid Range";
        return 0;
    }

    int nominal = (min + max) / 2;
    int values[] = {min, min + 1, nominal, max - 1, max};
```

```
    cout << "a\tb\tc\tOutput" << endl;

    for (int i = 0; i < 5; i++)
    {
        bva(values[i], nominal, nominal);
    }

    for (int i = 0; i < 5; i++)
    {
        if (values[i] != nominal)
            bva(nominal, values[i], nominal);
    }

    for (int i = 0; i < 5; i++)
    {
        if (values[i] != nominal)
            bva(nominal, nominal, values[i]);
    }

    cout << "Enter the Sides of Triangle (a, b, c) : ";
    cin >> x >> y >> z;
    cout << "a\tb\tc\tOutput" << endl;
    bva(x, y, z);

    return 0;
}

void bva(int a, int b, int c)
{
    cout << a << "\t" << b << "\t" << c << "\t";

    if (a < 1 || a > 100 || b < 1 || b > 100 || c < 1 || c > 100)
        cout << "Invalid Range" << endl;

    else if ((a < b + c) && (b < a + c) && (c < a + b))
    {
        if ((a == b) && (b == c))
            cout << "Equilateral Triangle" << endl;
        else if ((a != b) && (b != c) && (c != a))
            cout << "Scalene Triangle" << endl;
        else
            cout << "Isosceles Triangle" << endl;
    }

    else
        cout << "Not a Triangle" << endl;
}
```

Boundary Value Analysis :

Range : R [1, 100]

Domain : Minimum = 1

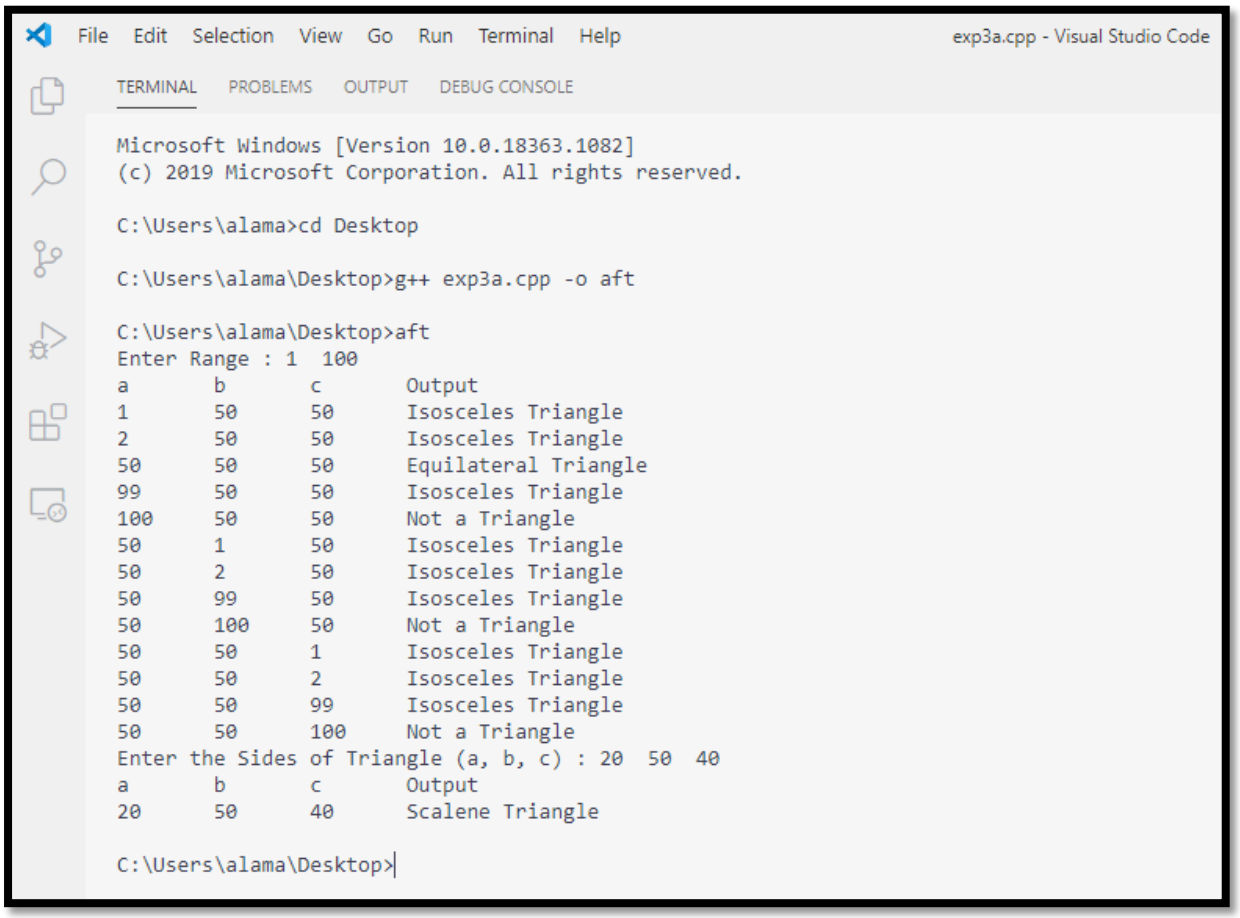
Above Minimum = 2

Nominal = 50

Below Maximum = 99

Maximum = 100

Output Screenshot :



Aim :

(b) Write a program for classification of Triangle on the basis of angle. Its input is triple of positive integers (say a, b, c) and values may be from the interval [1, 100]. The output may have one of the following :

Right Angled, Acute Angled, Obtuse Angled or Not a Triangle.

Perform Boundary Value Analysis and show the Robust Test Cases.

Algorithm :

- Take 3 inputs from the user for the sides of the Triangle.
- Check whether they lie in the given interval.
- If the condition is false, stop the program and exit.
- If the condition is true, check the type of Triangle.
- If the square of one side is equal to the sum of squares of other two sides, Right Angled Triangle.
- Else if the square of one side is greater than the sum of squares of other two sides, Obtuse Angled.
- Else if the square of one side is less than the sum of squares of other two sides, Acute Angled.
- According to the formula ***4 n + 1,*** there will be ***13*** test cases, where n is number of inputs.
- Robust Test Cases ***= 6 n + 1 = 6 * 3 + 1 = 18 + 1 = 19.***

Code :

```
#include <iostream>

using namespace std;

void bva(int, int, int);

int main()
{
    int min, max;
    int x, y, z;
    cout << "Enter Range : ";
    cin >> min >> max;

    if (min < 0 || max > 100)
    {
        cout << "Invalid Range";
        return 0;
    }

    int nominal = (min + max) / 2;
    int values[] = {min, min + 1, nominal, max - 1, max};
```

```
    cout << "a\tb\tc\tOutput" << endl;

    for (int i = 0; i < 5; i++)
    {
        bva(values[i], nominal, nominal);
    }

    for (int i = 0; i < 5; i++)
    {
        if (values[i] != nominal)
            bva(nominal, values[i], nominal);
    }

    for (int i = 0; i < 5; i++)
    {
        if (values[i] != nominal)
            bva(nominal, nominal, values[i]);
    }

    cout << "Enter the Sides of Triangle (a, b, c) : ";
    cin >> x >> y >> z;
    cout << "a\tb\tc\tOutput" << endl;
    bva(x, y, z);

    return 0;
}

void bva(int a, int b, int c)
{
    cout << a << "\t" << b << "\t" << c << "\t";

    if (a < 1 || a > 100 || b < 1 || b > 100 || c < 1 || c > 100)
        cout << "Invalid Range" << endl;

    else if ((a < b + c) && (b < a + c) && (c < a + b))
    {
        if ((a * a == b * b + c * c) || (b * b == c * c + a * a)
        || (c * c == a * a + b * b))
            cout << "Right Angled Triangle" << endl;
        else if ((a * a > b * b + c * c) || (b * b > c * c + a * a)
        || (c * c > a * a + b * b))
            cout << "Obtuse Angled Triangle" << endl;
        else
            cout << "Acute Angled Triangle" << endl;
    }
    else
        cout << "Not a Triangle" << endl;
}
```

Boundary Value Analysis :

Range : R [1, 100]

Domain : Below Minimum = 0

Minimum = 1

Above Minimum = 2

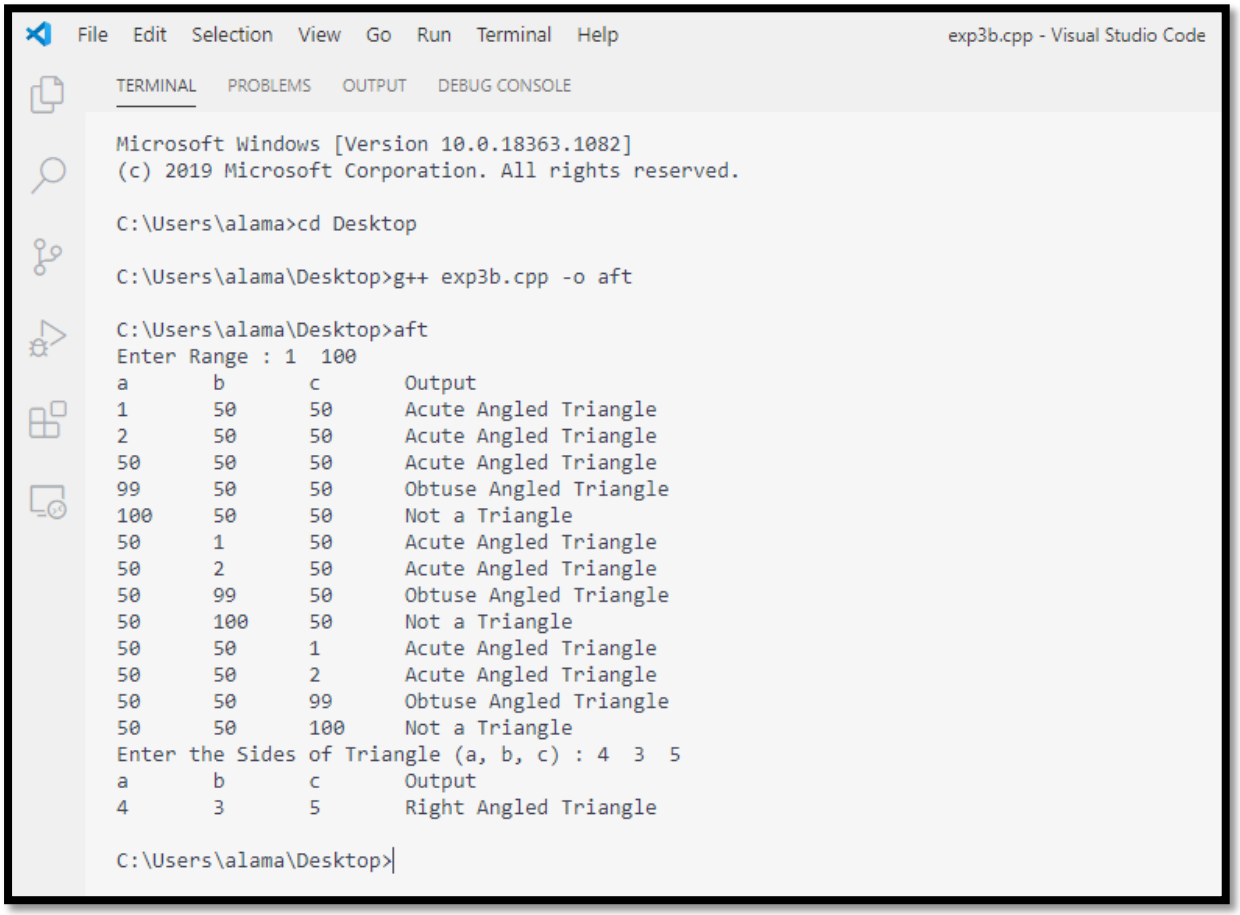
Nominal = 50

Below Maximum = 99

Maximum = 100

Above Maximum = 101

Output Screenshot :



Mohd. Aftab Alam
02013302717

Robust Test Cases :

Test Case	a	b	c	Expected Output
1	1	50	50	Acute Angled Triangle
2	2	50	50	Acute Angled Triangle
3	50	50	50	Acute Angled Triangle
4	99	50	50	Obtuse Angled Triangle
5	100	50	50	Not a Triangle
6	0	50	50	Invalid Range
7	101	50	50	Invalid Range
8	50	1	50	Acute Angled Triangle
9	50	2	50	Acute Angled Triangle
10	50	99	50	Obtuse Angled Triangle
11	50	100	50	Not a Triangle
12	50	0	50	Invalid Range
13	50	101	50	Invalid Range
14	50	50	1	Acute Angled Triangle
15	50	50	2	Acute Angled Triangle
16	50	50	99	Obtuse Angled Triangle
17	50	50	100	Not a Triangle
18	50	50	0	Invalid Range
19	50	50	101	Invalid Range