Lab Assignment 8

IT-314

Name : Goswami Jenil

Id : 202201247

Q1 :Consider a program for determining the previous date. Its input is triple of day, month and year with the following ranges 1 <= month <= 12, 1 <= day <= 31, 1900 <= year <= 2015.The possible output dates would be previous date or invalid date. Design the equivalence class test cases?

1. Program Specification

Input: Triple of day, month, and year

Input ranges:

1 <= month <= 12

1 <= day <= 31

1900 <= year <= 2015

Output: Previous date or "Invalid date"

2.Test

2.1 Valid Date Inputs:

- Any date that is valid according to the rules.
- For example: (1, 1, 1900), (31, 12, 2015), (29, 2, 2012) (leap year).

2.2 Invalid Date Inputs:

- Dates that do not exist or are out of the specified ranges.
- Invalid month (< 1 or > 12)
- Invalid day (< 1 or > max days in month)

- Invalid year (< 1900 or > 2015) (31, 4, 2015) → April has 30 days.
- (32, 1, 2015) → Day exceeds 31.

3. Boundary Value Analysis (BVA)

3.1 Lower Boundaries:

- (1, 1, 1900) : Valid lower boundary date.
- (1, 1, 1899) : Invalid year (lower than 1900).
- (1, 1, 2015) : Valid upper boundary year.

3.2 Upper Boundaries:

- (31, 12, 2015) → Valid upper boundary date.
- (1, 12, 2015) → Edge case for December.
- (29, 2, 2004) → Valid leap year case.

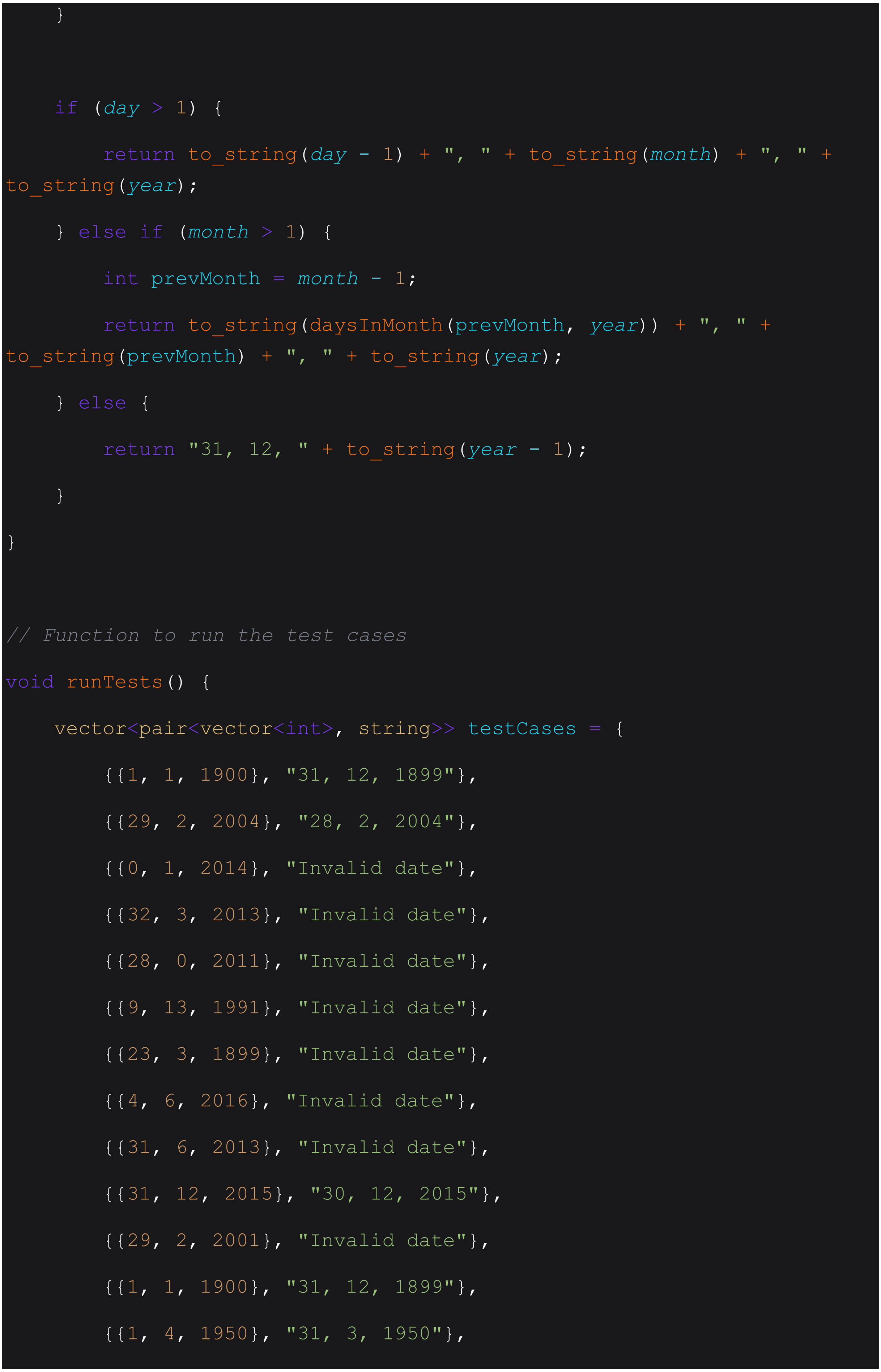
Equivalence Partitioning Test Cases

Tester Action and Input Data	Expected Outcome	Remarks
x,y,z	An Error Message	Invalid input
1,1,1900	31,12,1899	Boundary: Minimum valid year
29,2,2004	28,2,2004	leap year
0,1,2014	Invalid date	Day < 1
32,3,2013	Invalid date	Day > 31
28,0,2011	Invalid date	Month < 1
9,13,1991	Invalid date	Month > 12
23,3,1989	Invalid date	Year < 1900
4,6,2016	Invalid date	Year > 2015
31,6,2013	Invalid date	Invalid day for June
31,12,2015	30, 12, 2015	Boundary: Maximum valid year
29, 2, 2001	31, 12, 1899	Boundary: Minimum valid year
28,2,2006	27,2,2006	Boundary: Last day of February in non-leap year
1, 1, 1900	31, 12, 1899	Boundary: First day of year
31, 12, 1899	30, 12, 1910	Boundary: Last day of year
1, 4, 1950	31, 3, 1950	Boundary: First day of month
31, 7, 1980	30, 7, 1980	Boundary: Last day of 31-day month
30, 9, 1995	29, 9, 1995	Boundary: Last day of 30-day month

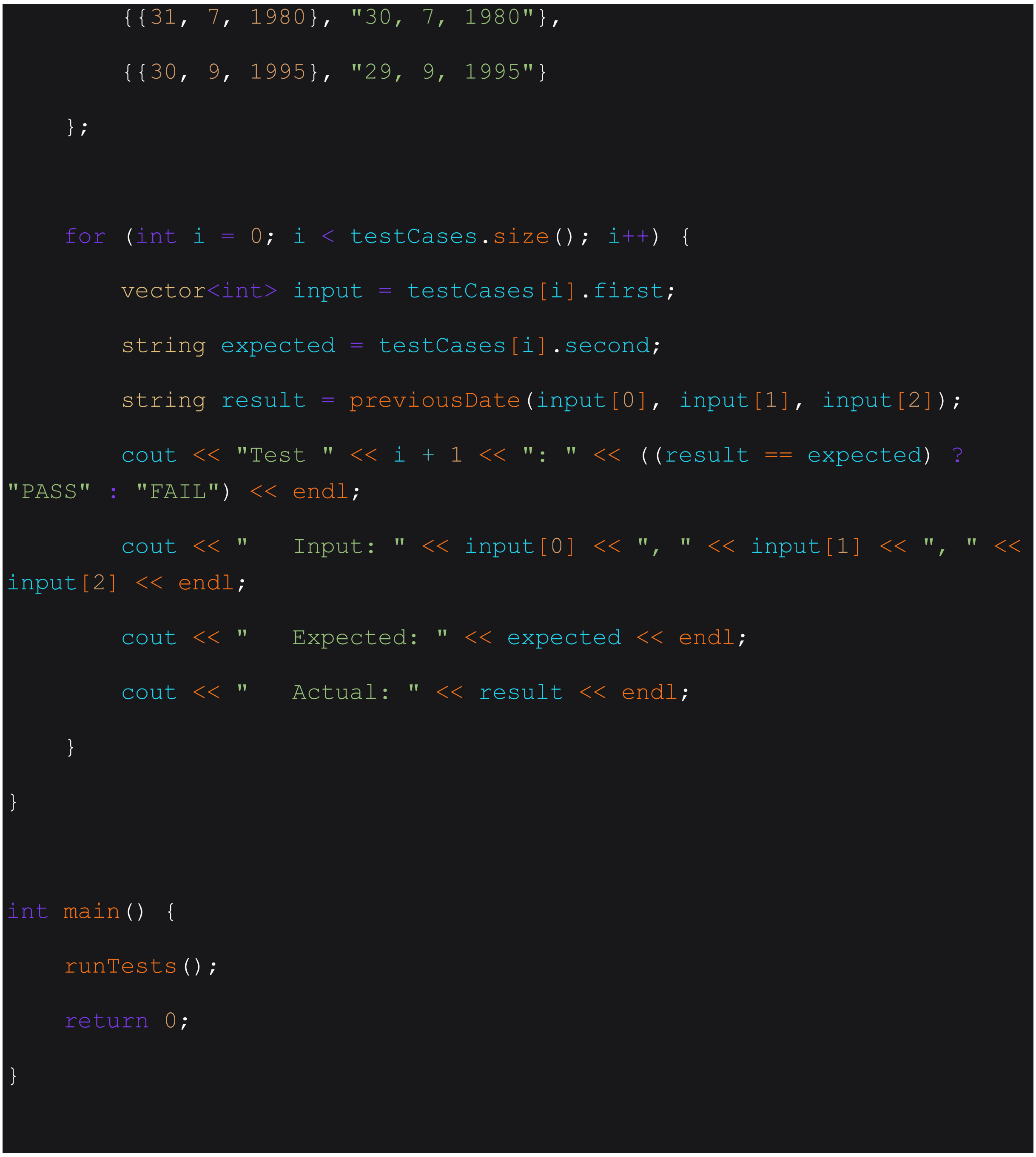
c++ implementation :

```
#include <iostream>
```

```
#include <vector>
#include <string>

using namespace std;

// Function to check if a year is a leap year
bool isLeapYear(int year) {
    return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
}

// Function to get the number of days in a given month of a given year
int daysInMonth(int month, int year) {
    vector<int> days = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30,
31};
    if (month == 2 && isLeapYear(year)) {
        return 29;
    }
    return days[month - 1];
}

// Function to calculate the previous date
string previousDate(int day, int month, int year) {
    if (!(1 <= month && month <= 12 && 1 <= year && year >= 1900 &&
year <= 2015)) {
        return "Invalid date";
    }

    int maxDays = daysInMonth(month, year);
    if (!(1 <= day && day <= maxDays)) {
        return "Invalid date";
```



```
    }


    if (day > 1) {
        return to_string(day - 1) + ", " + to_string(month) + ", " +
to_string(year);
    } else if (month > 1) {
        int prevMonth = month - 1;
        return to_string(daysInMonth(prevMonth, year)) + ", " +
to_string(prevMonth) + ", " + to_string(year);
    } else {
        return "31, 12, " + to_string(year - 1);
    }
}


// Function to run the test cases
void runTests() {
    vector<pair<vector<int>, string>> testCases = {
        {{1, 1, 1900}, "31, 12, 1899"},
        {{29, 2, 2004}, "28, 2, 2004"},
        {{0, 1, 2014}, "Invalid date"},
        {{32, 3, 2013}, "Invalid date"},
        {{28, 0, 2011}, "Invalid date"},
        {{9, 13, 1991}, "Invalid date"},
        {{23, 3, 1899}, "Invalid date"},
        {{4, 6, 2016}, "Invalid date"},
        {{31, 6, 2013}, "Invalid date"},
        {{31, 12, 2015}, "30, 12, 2015"},
        {{29, 2, 2001}, "Invalid date"},
        {{1, 1, 1900}, "31, 12, 1899"},
        {{1, 4, 1950}, "31, 3, 1950"},
```



```
        {{31, 7, 1980}, "30, 7, 1980"},
        {{30, 9, 1995}, "29, 9, 1995"}
    };

    for (int i = 0; i < testCases.size(); i++) {
        vector<int> input = testCases[i].first;
        string expected = testCases[i].second;
        string result = previousDate(input[0], input[1], input[2]);
        cout << "Test " << i + 1 << ": " << ((result == expected) ? "PASS" : "FAIL") << endl;
        cout << "   Input: " << input[0] << ", " << input[1] << ", " << input[2] << endl;
        cout << "   Expected: " << expected << endl;
        cout << "   Actual: " << result << endl;
    }
}

int main() {
    runTests();
    return 0;
}
```

Q2
P1 :

Equivalence Partitioning

Test Case	**Input v**	**Input a**	**Expected Output**	**Remarks**
TC1	5	{1, 2, 5, 7}	2	Value v exists
TC2	8	{1, 2, 5, 7}	-1	Value v not found
TC3	3	{}	-1	Empty array
TC4	5	{1, 5, 5, 7}	1	v appears multiple times
TC5	5	{5}	0	Single-element array, v exists
TC6	3	{5}	-1	Single-element array,v not found

Boundary Value Analysis

Test Case	Input v	Input a	Expected Output	Remarks
B1	1	{1,2,3,4}	0	v is the first element
B2	4	{1,2,3,4}	3	v is the last element
B3	5	{5}	0	Array has one element, v exists
B4	3	{5}	-1	Array has one element, v not found
B5	1	{}	-1	Empty array
B6	100	{1, 2, ..., 100, 101}	99	Large array, v near the end

P2 :

Equivalence Partitioning

Test Case	Input v	Input a	Expected Output	Remarks
TC1	3	[1, 2, 3, 3, 4]	2	Value is present multiple times.
TC2	5	[1, 2, 3, 4]	0	Value is absent in the array.
TC3	1	[]	0	Array is empty.
TC4	2	[1, 2, 3]	1	Value is present once.
TC5	1	[1, 1, 1]	3	All elements are the target value.

Boundary Value Analysis

Test Case	Input v	Input a	Expected Output	Remarks
TC1	0	[]	0	Testing empty array.
TC2	1	[2]	0	Array size is 1, value not present.
TC3	2	[2]	1	Array size is 1, value present.
TC4	3	[3, 4]	1	Array size is 2, value present once.

TC5	5	[5, 5]	2	Array size is 2, value present twice.
TC6	6	[1, 2, 3, ..., 6, ..., 6]	Count of 6s	Testing large array with multiple 6s.
TC7	7	[1, 2, 3, ..., 6]	0	Testing large array with absent value.

P3
Equivalence Partitioning

Test Case	Input v	Input a	Expected Output	Remarks
TC1	3	[1, 2, 3, 4, 5]	2	Value is present in the array (index 2).
TC2	6	[1, 2, 3, 4, 5]	-1	Value is absent in the array.
TC3	1	[1, 2, 3, 4, 5]	0	Value is the first element (index 0).
TC4	5	[1, 2, 3, 4, 5]	4	Value is the last element (index 4).
TC5	4	[1, 2, 3, 4, 5]	3	Value is present in the middle (index 3).

TC6	0	[1, 2, 3, 4, 5]	-1	Value is less than all elements.
TC7	10	[1, 2, 3, 4, 5]	-1	Value is greater than all elements.
TC8	1	[1, 1, 1, 1, 1]	0	Value is present multiple times (first index).

Boundary Value Analysis

Test Case	**Input v**	**Input a**	**Expected Output**	**Remarks**
TC1	1	[1, 2, 3, 4, 5]	0	First element (lower boundary).
TC2	5	[1, 2, 3, 4, 5]	4	Last element (upper boundary).
TC3	3	[1, 2, 3, 4, 5]	2	Middle element.
TC4	0	[1, 2, 3, 4, 5]	-1	Value less than the smallest element.
TC5	6	[1, 2, 3, 4, 5]	-1	Value greater than the largest element.
TC6	1	[1]	0	Single-element array, matching value.

TC7	2	[1]	-1	Single-element array, non-matching value.
TC8	1	[]	-1	Empty array (no elements).
TC9	5	[]	-1	Searching in an empty array.

P4
Equivalence Partitioning

Test Case	Input a	Input b	Input c	Expected Output	Remarks
TC1	3	3	3	0	Equilateral triangle (all sides equal).
TC2	3	3	4	1	Isosceles triangle (two sides equal).
TC3	3	4	5	2	Scalene triangle (no sides equal).
TC4	1	1	2	3	Invalid triangle (not possible).
TC5	1	2	3	3	Invalid triangle (not possible).

TC6	0	0	0	3	Invalid triangle (zero lengths).
TC7	-1	-1	-1	3	Invalid triangle (negative lengths).
TC8	4	5	6	2	Scalene triangle (no sides equal).
TC9	5	5	8	1	Isosceles triangle (two sides equal).

Boundary Value Analysis

Test Case	**Input a**	**Input b**	**Input c**	**Expected Output**	**Remarks**
TC1	1	1	1	0	Minimum valid triangle (equilateral).
TC2	2	2	3	1	Minimum isosceles triangle.
TC3	3	4	5	2	Minimum scalene triangle.
TC4	1	1	2	3	Minimum invalid triangle.
TC5	0	1	1	3	Invalid triangle (zero length).
TC6	2	2	5	1	Isosceles triangle (two sides equal).

TC7	1	2	3	3	Invalid triangle (not possible).
TC8	1	1	0	3	Invalid triangle (zero length).
TC9	0	0	0	3	Invalid triangle (all sides zero).
TC10	-1	-1	-1	3	Invalid triangle (negative lengths).

P5
Equivalence Partitioning

Test Case	**Input s1**	**Input s2**	**Expected Output**	**Remarks**
TC1	"pre"	"prefix"	TRUE	s1 is a prefix of s2.
TC2	"prefix"	"prefix"	TRUE	s1 is equal to s2 (s1 is a prefix).
TC3	"prefix"	"pre"	FALSE	s1 is longer than s2.
TC4	"pre"	"pre"	TRUE	s1 is equal to s2 (both strings are equal).
TC5	"ab"	"abc"	TRUE	s1 is a prefix of s2.
TC6	"abc"	"ab"	FALSE	s1 is longer than s2.

TC7	""	"prefix"	TRUE	Empty string s1 is a prefix of any s2.
TC8	"prefix"	""	FALSE	s1 is longer than s2 (s2 is empty).
TC9	""	""	TRUE	Both strings are empty.

Boundary Value Analysis

Test Case	**Input s1**	**Input s2**	**Expected Output**	**Remarks**
TC1	""	""	TRUE	Both strings are empty (base case).
TC2	"a"	"a"	TRUE	Both strings are equal (single character).
TC3	"a"	"ab"	TRUE	s1 is a prefix of s2 (single character).
TC4	"ab"	"abc"	TRUE	s1 is a prefix of s2 (two characters).
TC5	"abc"	"ab"	FALSE	s1 is longer than s2 (invalid case).

TC6	"prefix"	"prefixes"	TRUE	s1 is a prefix of s2 (longer s2).
TC7	"prefixes"	"prefix"	FALSE	s1 is longer than s2 (invalid case).
TC8	"p"	"prefix"	TRUE	Single character prefix.
TC9	"pre"	"pre"	TRUE	Both strings are equal.
TC10	"longprefix"	"prefix"	FALSE	s1 is longer than s2 (invalid case).

P6

a) Identify the Equivalence Classes

1. **Equilateral Triangle**: All three sides are equal ($A = B = C$).
2. **Isosceles Triangle**: Exactly two sides are equal ($A = B \neq C$, $A = C \neq B$, $B = C \neq A$).
3. **Scalene Triangle**: All sides are different ($A \neq B \neq C$).
4. **Right-Angled Triangle**: Fulfills Pythagorean theorem ($A^2 + B^2 = C^2$ or any permutation).
5. **Invalid Triangle**: Cannot form a triangle ($A + B \leq C$, $A + C \leq B$, $B + C \leq A$).
6. **Non-Triangle**: One or more sides are non-positive ($A \leq 0$, $B \leq 0$, $C \leq 0$).

b) Identify Test Cases for Equivalence Classes

Test Case	**Input A**	**Input B**	**Input C**	**Expected Output**	**Equivalence Class Covered**
TC1	3	3	3	"Equilateral"	Equilateral Triangle

TC2	3	3	4	"Isosceles"	Isosceles Triangle
TC3	3	4	5	"Scalene"	Scalene Triangle
TC4	5	12	13	"Right-Angled"	Right-Angled Triangle
TC5	1	2	3	"Invalid"	Invalid Triangle
TC6	2	2	5	"Invalid"	Invalid Triangle
TC7	-1	2	3	"Non-Triangle"	Non-Triangle
TC8	0	2	2	"Non-Triangle"	Non-Triangle

c) Boundary Condition for A + B > C (Scalene Triangle)

Test Case	**Input A**	**Input B**	**Input C**	**Expected Output**	**Remarks**
BC1	1	1	1.9	"Scalene"	Valid triangle (A + B > C).
BC2	1	1	2	"Invalid"	Boundary case (A + B = C).
BC3	1	2	1.9	"Scalene"	Valid triangle (A + B > C).
BC4	2	2	4	"Invalid"	Invalid triangle (A + B = C).

d) Boundary Condition for A = C (Isosceles Triangle)

Test Case	**Input A**	**Input B**	**Input C**	**Expected Output**	**Remarks**
BC1	3	4	3	"Isosceles"	Valid isosceles triangle.
BC2	3	3	3	"Equilateral"	Edge case (all sides equal).

BC3	4	5	4	"Isosceles"	Valid isosceles triangle.
BC4	1	1	1	"Equilateral"	Edge case (all sides equal).

e) Boundary Condition for A = B = C (Equilateral Triangle)

Test Case	Input A	Input B	Input C	Expected Output	Remarks
BC1	2	2	2	"Equilateral"	All sides equal (valid).
BC2	3	3	3	"Equilateral"	All sides equal (valid).
BC3	1	1	1	"Equilateral"	Minimum valid triangle.

f) Boundary Condition for $A^2 + B^2 = C^2$ (Right-Angled Triangle)

Test Case	Input A	Input B	Input C	Expected Output	Remarks
BC1	3	4	5	"Right-Angled"	Valid right-angled triangle.
BC2	5	12	13	"Right-Angled"	Valid right-angled triangle.
BC3	6	8	10	"Right-Angled"	Valid right-angled triangle.
BC4	1	1	1.5	"Invalid"	Not a right-angled triangle (not valid).

g) Non-Triangle Case

Test Case	Input A	Input B	Input C	Expected Output	Remarks
NC1	1	1	3	"Invalid"	A + B ≤ C (invalid triangle).
NC2	2	3	6	"Invalid"	A + B ≤ C (invalid triangle).
NC3	1	2	2	"Invalid"	A + B ≤ C (invalid triangle).
NC4	2	2	5	"Invalid"	A + B ≤ C (invalid triangle).

h) Non-Positive Input

Test Case	Input A	Input B	Input C	Expected Output	Remarks
NP1	0	1	1	"Non-Triangle"	One side is zero (invalid).
NP2	1	0	1	"Non-Triangle"	One side is zero (invalid).
NP3	1	1	0	"Non-Triangle"	One side is zero (invalid).
NP4	-1	1	1	"Non-Triangle"	One side is negative (invalid).
NP5	1	-1	1	"Non-Triangle"	One side is negative (invalid).
NP6	1	1	-1	"Non-Triangle"	One side is negative (invalid).