IT314
SOFTWARE ENGINEERING

LAB-08
Software Testing
Lab Session- Functional Testing (Black-Box)

Name: SMIT FEFAR

ID: 202201253

Q.1. 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?

Answer:

Equivalence Classes:

- E1: Valid Day (1 <= day <= 31)
- E2: Day<1 (Invalid)
- E3: Day>31 (Invalid)
- E4: 1<=month<=12 (Valid month)
- E5: month <1 (Invalid)
- E6: month >12 (Invalid)
- E7: Valid Year (1900 <= year <= 2015)
- E8: Year <1900 (Invalid)
- E9: Year >2015 (Invalid)

Test cases: (For invalid and valid inputs):

Test case(D,M,Y)	Classes covered	Expected Output
(2,2,1901)	E1, E4, E7	Valid
(0, 1, 2016)	E2, E4, E9	Invalid
(13, 13, 2010)	E1, E4, E7	Invalid
(32,1, 2010)	E3, E4, E7	Invalid
10,12,2000	E1, E4, E7	Valid

Write a set of test cases (i.e., test suite) – specific set of data – to properly test the programs. Your test suite should include both correct and incorrect inputs.

1. Enlist which set of test cases have been identified using Equivalence Partitioning and Boundary Value Analysis separately.

2. Modify your programs such that it runs, and then execute your test suites on the program.

While executing your input data in a program, check whether the identified expected outcome (mentioned by you) is correct or not.

Programme: 1

Test Suite

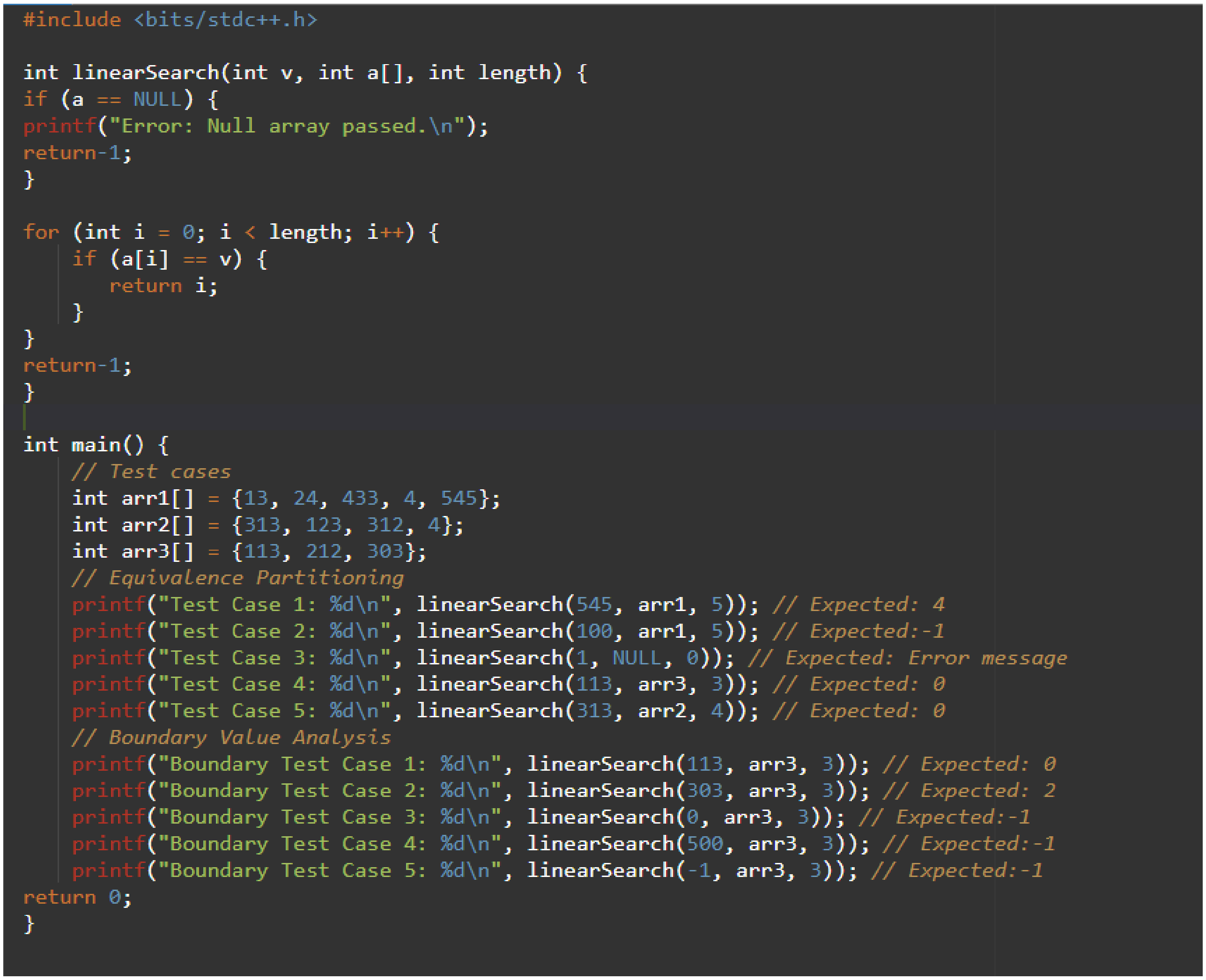
Equivalence Partitioning

Testing and Input Data (number, array)	Expected Output (index)
linearSearch(5, [4, 5])	4
linearSearch(6, [1, 2, 3, 4, 5])	-1
linearSearch(1, [])	-1
linearSearch(3.5, [1, 2, 3])	An Error message
linearSearch(1, NULL)	An Error message
linearSearch(3, [3, 3, 4])	0

Boundary Value Analysis

Testing and Input Data (number, array)	Expected Output (index)
linearSearch(3, [2, 3,545])	1
linearSearch(1, [1, 2, 3,4])	0
linearSearch(4, [1, 2, 3])	-1
linearSearch(-1, [1, 2, 3])	-1

linearSearch Code:



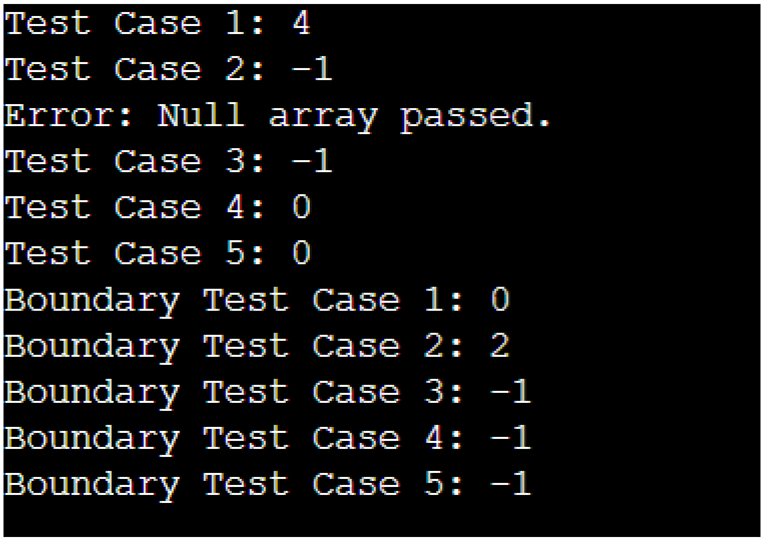
```
#include <bits/stdc++.h>

int linearSearch(int v, int a[], int length) {
if (a == NULL) {
printf("Error: Null array passed.\n");
return-1;
}

for (int i = 0; i < length; i++) {
    if (a[i] == v) {
        return i;
    }
}
return-1;
}

int main() {
    // Test cases
    int arr1[] = {13, 24, 433, 4, 545};
    int arr2[] = {313, 123, 312, 4};
    int arr3[] = {113, 212, 303};
    // Equivalence Partitioning
    printf("Test Case 1: %d\n", linearSearch(545, arr1, 5)); // Expected: 4
    printf("Test Case 2: %d\n", linearSearch(100, arr1, 5)); // Expected:-1
    printf("Test Case 3: %d\n", linearSearch(1, NULL, 0)); // Expected: Error message
    printf("Test Case 4: %d\n", linearSearch(113, arr3, 3)); // Expected: 0
    printf("Test Case 5: %d\n", linearSearch(313, arr2, 4)); // Expected: 0
    // Boundary Value Analysis
    printf("Boundary Test Case 1: %d\n", linearSearch(113, arr3, 3)); // Expected: 0
    printf("Boundary Test Case 2: %d\n", linearSearch(303, arr3, 3)); // Expected: 2
    printf("Boundary Test Case 3: %d\n", linearSearch(0, arr3, 3)); // Expected:-1
    printf("Boundary Test Case 4: %d\n", linearSearch(500, arr3, 3)); // Expected:-1
    printf("Boundary Test Case 5: %d\n", linearSearch(-1, arr3, 3)); // Expected:-1
return 0;
}
```

Output:



```
Test Case 1: 4
Test Case 2: -1
Error: Null array passed.
Test Case 3: -1
Test Case 4: 0
Test Case 5: 0
Boundary Test Case 1: 0
Boundary Test Case 2: 2
Boundary Test Case 3: -1
Boundary Test Case 4: -1
Boundary Test Case 5: -1
```

Programme-2:

Test Suite:

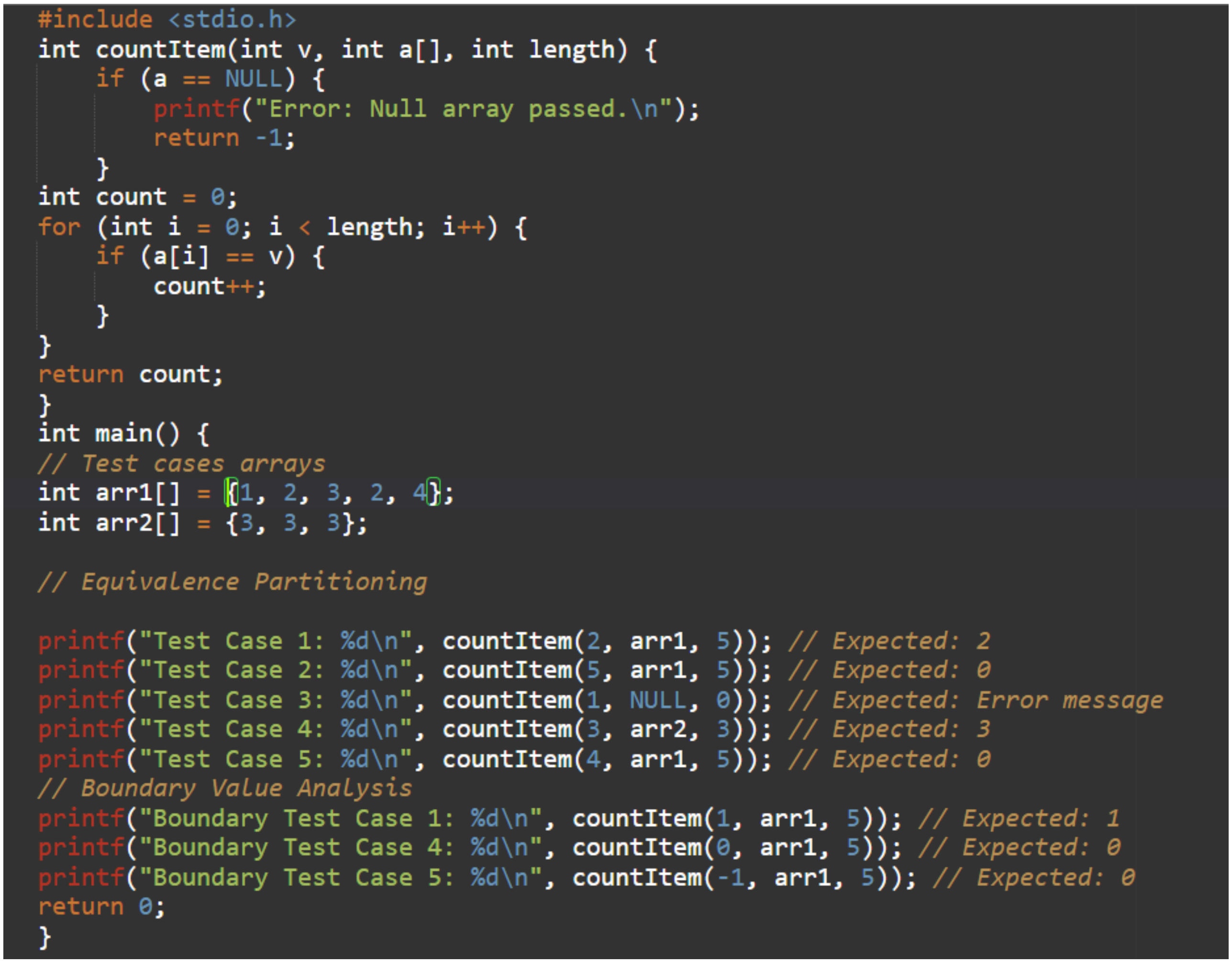
Equivalence Partitioning

Testing and Input Data (Item No, Array)	Expected Output (count)
countItem(2, [1, 2, 3, 4])	1
countItem(5, [1, 2, 3, 2, 4])	0
countItem(1, [])	0
countItem(3, [3, 3, 3])	3
countItem(2.5, [1, 2, 3, 2, 4])	An Error message
countItem(1, NULL)	An Error message

Boundary Value Analysis

Testing and Input Data (Item No, Array)	Expected Output (count)
countItem(1, [1, 2, 3])	1
countItem(3, [1, 2, 3])	1
countItem(0, [1, 2, 3])	0
countItem(-1, [1, 2, 3])	0

CountItem Code:

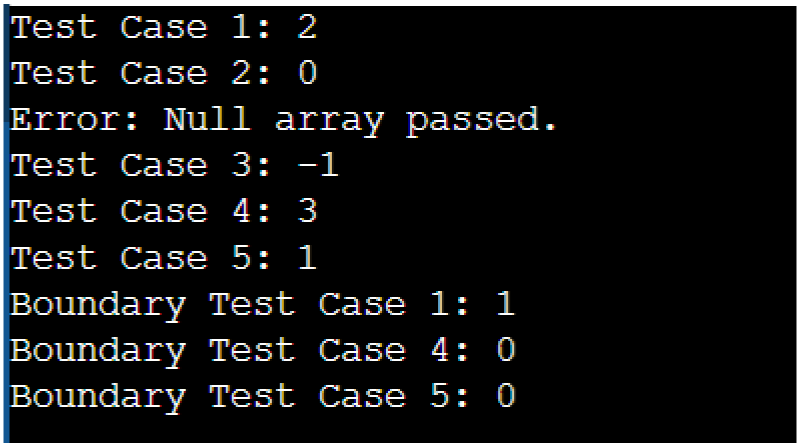


```c
#include <stdio.h>
int countItem(int v, int a[], int length) {
    if (a == NULL) {
        printf("Error: Null array passed.\n");
        return -1;
    }
int count = 0;
for (int i = 0; i < length; i++) {
    if (a[i] == v) {
        count++;
    }
}
return count;
}
int main() {
// Test cases arrays
int arr1[] = {1, 2, 3, 2, 4};
int arr2[] = {3, 3, 3};

// Equivalence Partitioning

printf("Test Case 1: %d\n", countItem(2, arr1, 5)); // Expected: 2
printf("Test Case 2: %d\n", countItem(5, arr1, 5)); // Expected: 0
printf("Test Case 3: %d\n", countItem(1, NULL, 0)); // Expected: Error message
printf("Test Case 4: %d\n", countItem(3, arr2, 3)); // Expected: 3
printf("Test Case 5: %d\n", countItem(4, arr1, 5)); // Expected: 0
// Boundary Value Analysis
printf("Boundary Test Case 1: %d\n", countItem(1, arr1, 5)); // Expected: 1
printf("Boundary Test Case 4: %d\n", countItem(0, arr1, 5)); // Expected: 0
printf("Boundary Test Case 5: %d\n", countItem(-1, arr1, 5)); // Expected: 0
return 0;
}
```

Output:



```
Test Case 1: 2
Test Case 2: 0
Error: Null array passed.
Test Case 3: -1
Test Case 4: 3
Test Case 5: 1
Boundary Test Case 1: 1
Boundary Test Case 4: 0
Boundary Test Case 5: 0
```

Programme: 3

Test Suite:

Equivalence Partitioning:

Testing and Input Data (Item No, Array)	Expected Output (index)
binarySearch(3, [1, 2, 3, 4, 5])	2
binarySearch(6, [3, 2, 1, 4, 5])	An invalid message
binarySearch(1, [1, 2, 3, 4, 5])	0
binarySearch(5, [1, 2, 3, 4, 5])	4
binarySearch(3.5, [1, 2, 3, 4, 5])	An invalid message
binarySearch(1, NULL)	An invalid message

Boundary Value Analysis:

Testing and Input Data (Item No, Array)	Expected Output (count)
binarySearch(3, [1, 2, 3])	2
binarySearch(2, [1, 2, 3])	1
binarySearch(0, [1, 2, 3])	-1
binarySearch(4, [1, 2, 3])	-1

binarySearch Code:

```
#include <stdio.h>
int binarySearch(int v, int a[], int length) {
    if (a == NULL) {
    printf("Error: Null array passed.\n");
    return -1;
    }
    int lo = 0, hi = length - 1;
    while (lo <= hi) {
        int mid = (lo + hi) / 2;
        if (v == a[mid]) {
            return mid;
        }
        else if (v < a[mid]) {
            hi = mid - 1;
        } else {
            lo = mid + 1;
        }
    }
    return -1;
}
int main() {
// Test cases
int arr1[] = {1, 2, 3, 4, 5};
int arr2[] = {1, 2, 3};
// Equivalence Partitioning
printf("Test Case 1: %d\n", binarySearch(0, arr1, 5)); // Expected: -1
printf("Test Case 2: %d\n", binarySearch(6, arr1, 5)); // Expected: -1
printf("Test Case 3: %d\n", binarySearch(1, arr1, 5)); // Expected: 0
printf("Test Case 4: %d\n", binarySearch(5, arr1, 5)); // Expected: 4

printf("Test Case 5: %d\n", binarySearch(1, NULL, 0)); // Expected: Error
message

// Boundary Value Analysis
printf("Boundary Test Case 1: %d\n", binarySearch(1, arr2, 3)); // Expected:
0
printf("Boundary Test Case 2: %d\n", binarySearch(3, arr2, 3)); // Expected:
2
printf("Boundary Test Case 3: %d\n", binarySearch(2, arr2, 3)); // Expected:
```

```
1
printf("Boundary Test Case 4: %d\n", binarySearch(0, arr2, 3)); // Expected:
-1
printf("Boundary Test Case 5: %d\n", binarySearch(4, arr2, 3)); // Expected:
-1
return 0;
}
```

Output:

```
Test Case 1: -1
Test Case 2: -1
Test Case 3: 0
Test Case 4: 4
Error: Null array passed.
Test Case 5: -1
Boundary Test Case 1: 0
Boundary Test Case 2: 2
Boundary Test Case 3: 1
Boundary Test Case 4: -1
Boundary Test Case 5: -1
```

Programme: 4

Test Suite

Equivalence Partitioning

Testing and Input Data (side1,side2,side3)	Expected Output (type of triangle)
triangle(14, 14,14)	0 (EQUILATERAL)
triangle(3, 4, 3)	1 (ISOSCELES)
triangle(3, 4, 5)	2 (SCALENE)
triangle(0, 1, 1)	3 (INVALID)
triangle(-1, 2, 3)	3 (INVALID)
triangle(1.5, 1.5, 1.5)	invalid input

Boundary Value Analysis

Testing and Input Data (side1,side2,side3)	Expected Output (type of triangle)
triangle(15, 15, 15)	0 (EQUILATERAL)
triangle(3, 3, 4)	1 (ISOSCELES)
triangle(3, 4, 7)	3 (INVALID)

Triangle Code:

```
#include <stdio.h>
#define EQUILATERAL 0
#define ISOSCELES 1
#define SCALENE 2
#define INVALID 3
int triangle(int a, int b, int c) {
if (a <= 0 || b <= 0 || c <= 0 || a >= b + c || b >= a + c || c >= a + b) {
return INVALID;
}
if (a == b && b == c) {
return EQUILATERAL;
}
if (a == b || a == c || b == c) {
return ISOSCELES;
}
return SCALENE;
}

int main() {
// Test cases
printf("Test Case 1: %d\n", triangle(14, 14, 14));
printf("Test Case 2: %d\n", triangle(3, 4, 5));
printf("Test Case 3: %d\n", triangle(1, 1, 2));
printf("Test Case 4: %d\n", triangle(0, 1, 1));
printf("Test Case 5: %d\n", triangle(-1, 2, 3));

printf("Boundary Test Case 1: %d\n", triangle(1, 1, 1));
printf("Boundary Test Case 2: %d\n", triangle(2, 2, 3));
printf("Boundary Test Case 3: %d\n", triangle(3, 3, 4));
printf("Boundary Test Case 4: %d\n", triangle(2, 2, 5));
printf("Boundary Test Case 5: %d\n", triangle(3, 4, 7));
return 0;
}
```

Output:

```
Test Case 1: 0
Test Case 2: 2
Test Case 3: 3
Test Case 4: 3
Test Case 5: 3
Boundary Test Case 1: 0
Boundary Test Case 2: 1
Boundary Test Case 3: 1
Boundary Test Case 4: 3
Boundary Test Case 5: 3
```

Programme: 5

Test Suite for prefix

Equivalence Partitioning

Testing and Input Data (string1, string2)	Expected Output (Is it prefix or not?)
prefix("abc", "abcde")	true
prefix("test", "testing")	true
prefix("hello", "Gandhinagar")	false
prefix("wrong", "right")	false
prefix("", "nonempty")	true

Boundary Value Analysis

Testing and Input Data (string1, string2)	Expected Output (Is it prefix or not?)
prefix("", "")	true
prefix("ab", "a")	false
prefix("Hii", "hii")	False
prefix("Hello", "Hello")	true

prefix Code:

```
public class PrefixTest {
public static boolean prefix(String s1, String s2)
{
if (s1.length() > s2.length()) {
return false;
}
for (int i = 0; i < s1.length(); i++) {
if (s1.charAt(i) != s2.charAt(i)) {
return false;
}
}
return true;
}
}
public static void main(String[] args) {
// Test cases

System.out.println("Test Case 2: " + prefix("abc",
"abcdef"));
System.out.println("Test Case 3: " + prefix("test",
"testing"));
System.out.println("Test Case 4: " +
prefix("hello","Gandhinagar"));
System.out.println("Test Case 5: " +
prefix("wrong", "right"));
System.out.println("Test Case 6: " + prefix("",
"nonempty"));
System.out.println("Test Case 7: " +
prefix("nonempty", ""));

// Boundary Value Analysis

System.out.println("Boundary Test Case 1: " +
prefix("", ""));
System.out.println("Boundary Test Case 3: " +
prefix("ab", "a"));
System.out.println("Boundary Test Case 4: " +
prefix("Hii", "hii"));
System.out.println("Boundary Test Case 5: " +
prefix("Hello", "Hello"));
}
}
```

Programme-6:

a) Identify the equivalence classes for the system

Answer:

1. E1: Equilateral Triangle
   - (A = B = C).
2. E2: Isosceles Triangle
   - (A = B ≠ C, A = C ≠ B, B = C ≠ A).
3. E3: Scalene Triangle
   - (A ≠ B && B ≠ C && A ≠ C).

4. E4: Right-Angled Triangle
    - Satisfies Pythagorean theorem
5. E5: Invalid Triangle
    - Does not satisfy the triangle inequality
6. E6: Non-positive Input
    - (A ≤ 0 || B ≤ 0 || C ≤ 0).

b) Identify test cases to cover the identified equivalence classes. Also, explicitly mention which test case would cover which equivalence class. (Hint: you must need to be ensure that the identified set of test cases cover all identified equivalence classes)

Test Case	Side A	Side B	Side C	Expected Outcome	Equivalence Class
Test Case 1	14.0	14.0	14.0	Equilateral	E1 (Equilateral Triangle)
Test Case 2	3.0	3.0	5.0	Isosceles	E2 (Isosceles Triangle)
Test Case 3	3.0	7.0	5.0	Scalene	E3 (Scalene Triangle)
Test Case 4	3.0	4.0	6.0	Invalid	E5 (Invalid Triangle)
Test Case 5	3.0	4.0	5.0	Right-Angled	E4 (Right-Angled Triangle)
Test Case 6	0.0	0.0	5.0	Invalid	E6 (Non-positive Input)

c) For the boundary condition A + B > C case (scalene triangle), identify test cases to verify the boundary.

Test Case	Side A	Side B	Side C	Expected Outcome
Boundary Test Case 1	2.0	3.0	4.0	Scalene
Boundary Test Case 2	2.0	2.0	3.9	Scalene

d) For the boundary condition A = C case (isosceles triangle), identify test cases to verify the boundary.

Test Case	Side A	Side B	Side C	Expected Outcome
Boundary Test Case 1	3.0	4.0	3.0	Isosceles
Boundary Test Case 2	5.0	2.0	5.0	Isosceles

e) For the boundary condition A = B = C case (equilateral triangle), identify test cases to verify the boundary.

Test Case	Side A	Side B	Side C	Expected Outcome
Boundary Test Case 1	3.0	3.0	3.0	Equilateral
Boundary Test Case 2	5.0	5.0	5.0	Equilateral

f) For the boundary condition $A^2 + B^2 = C^2$ case (right-angle triangle), identify test cases to verify the boundary.

Test Case	Side A	Side B	Side C	Expected Outcome
Boundary Test Case 1	3.0	4.0	5.0	Right-Angled
Boundary Test Case 2	5.0	12.0	13.0	Right-Angled

g) For the non-triangle case, identify test cases to explore the boundary.

Test Case	Side A	Side B	Side C	Expected Outcome
Boundary Test Case 1	2.0	56.0	98.0	Invalid
Boundary Test Case 2	21.0	1.0	3.0	Invalid

h) For non-positive input, identify test points.

Test Case	Side A	Side B	Side C	Expected Outcome
Test Case 1	0.0	2.0	3.0	Invalid
Test Case 2	-1.0	0.0	3.0	Invalid