Lab 7 Report

Submitted in fulfilment of the course

IT314 (Software Engineering)

By

MAHARTH THAKAR

ID: 202001069

Under the guidance of

Prof. Saurabh Tiwari

12 Apr 2023

THE MAIN FILE

```
        package tests;
public class unittesting {
//      public int square(int n)
//      {
//              return n*n;
//      }

        public static int linearSearch(int v, int[] a) {
            int i = 0;
            while (i < a.length) {
                if (a[i] == v) {
                    return i;
                }
                i++;
            }
            return -1;
        }

        public static int countItem(int v, int[] a) {
            int count = 0;
            for (int i = 0; i < a.length; i++) {
                if (a[i] == v) {
                    count++;
                }
            }
            return count;
        }

        public static int binarySearch(int v, int[] a) {
            int lo = 0;
            int hi = a.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;
        }

        public static final int EQUILATERAL = 0;
        public static final int ISOSCELES = 1;
```

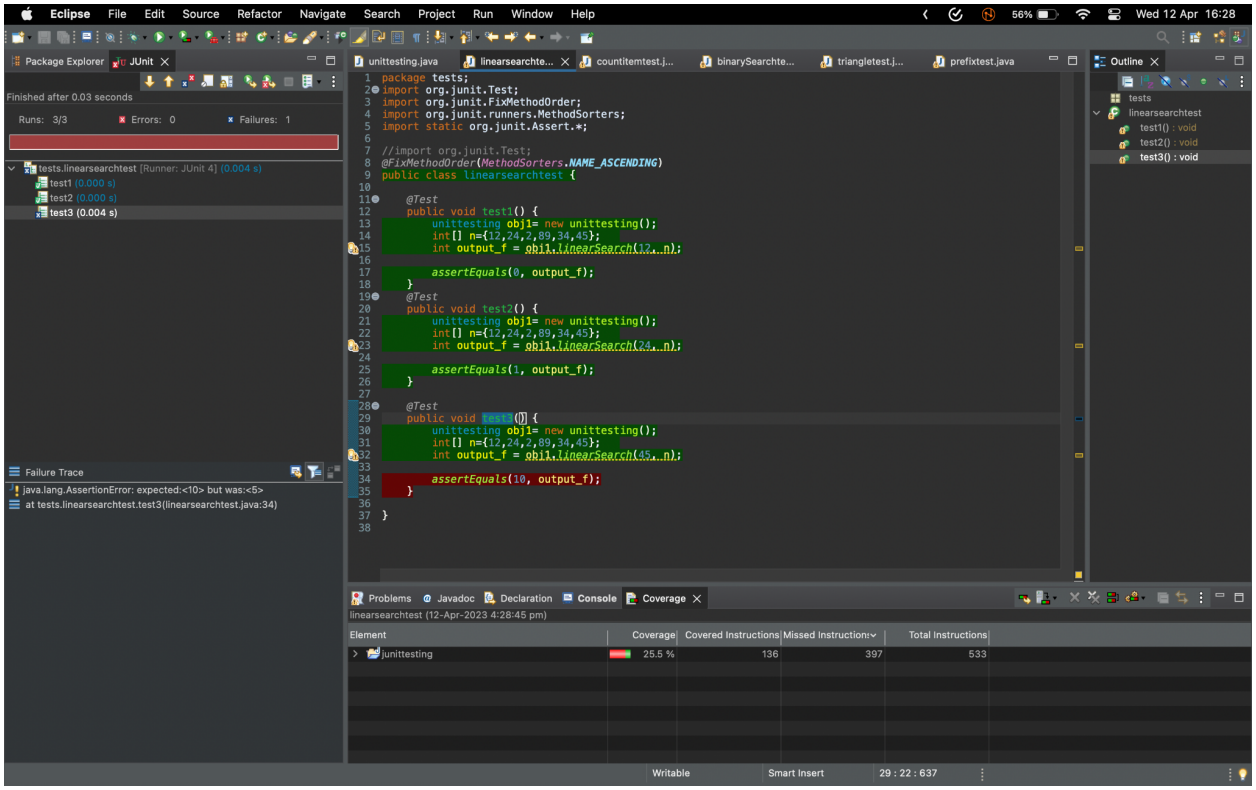
```
        public static final int SCALENE = 2;
        public static final int INVALID = 3;
        public static int triangle(int a, int b, int c) {
            if (a >= b + c || b >= a + c || c >= a + b) {
                return INVALID;
            }
            if (a == b && b == c) {
                return EQUILATERAL;
            }
            if (a == b || a == c || b == c) {
                return ISOSCELES;
            }
            return SCALENE;
        }


        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 int triangleP6(float a, float b, float c) {
//            if (a >= b + c || b >= a + c || c >= a + b) {
//                System.out.println("INVALID");
//            }
//            if (a == b && b == c) {
//                System.out.println("EQUILATERAL");
//                return EQUILATERAL;
//            }
//            if (a == b || a == c || b == c) {
//                System.out.println("ISOSCELES");
//                return ISOSCELES;
//            }
//            System.out.println("SCALENE");
//        }
}
```

Equivalence Partitioning : EP
Boundary Value Analysis : BVA

PROGRAM		Tester Action and Input Data	Expected Outcome
P1	EP	`int[] n={12,24,2,89,34,45};` ,24	1
	BVA	`int[] n={12,24,2,89,34,45};` ,12	0
	BVA	`int[] n={12,24,2,89,34,45};` ,45	5
P2	BVA	`int[] n={12,24,2,89,34,45};` ,12	1
	EP	`int[] n={12,24,2,89,34,45};` ,24	1
	EP	`int[] n={12,24,2,89,89,4`	2

		5}; ,89	
P3	EP	int[] n={1,2,3,4,5,6,7,8}; ,12	-1
	EP	int[] n={11,12,13,14,15, 16,17}; ,14	3
	BVA	int[] n={11,12,13,14,15, 16,17}; ,11	0
P4	EP	1,1,1	0
	EP	4,4,2	1
	BVA	1,2,3	3
	EP	12,5,13	2
P5	EP	"maharth", "maharththakar"	true
	EP	"hihello","hi"	false
	EP	"hello","helluhow"	false

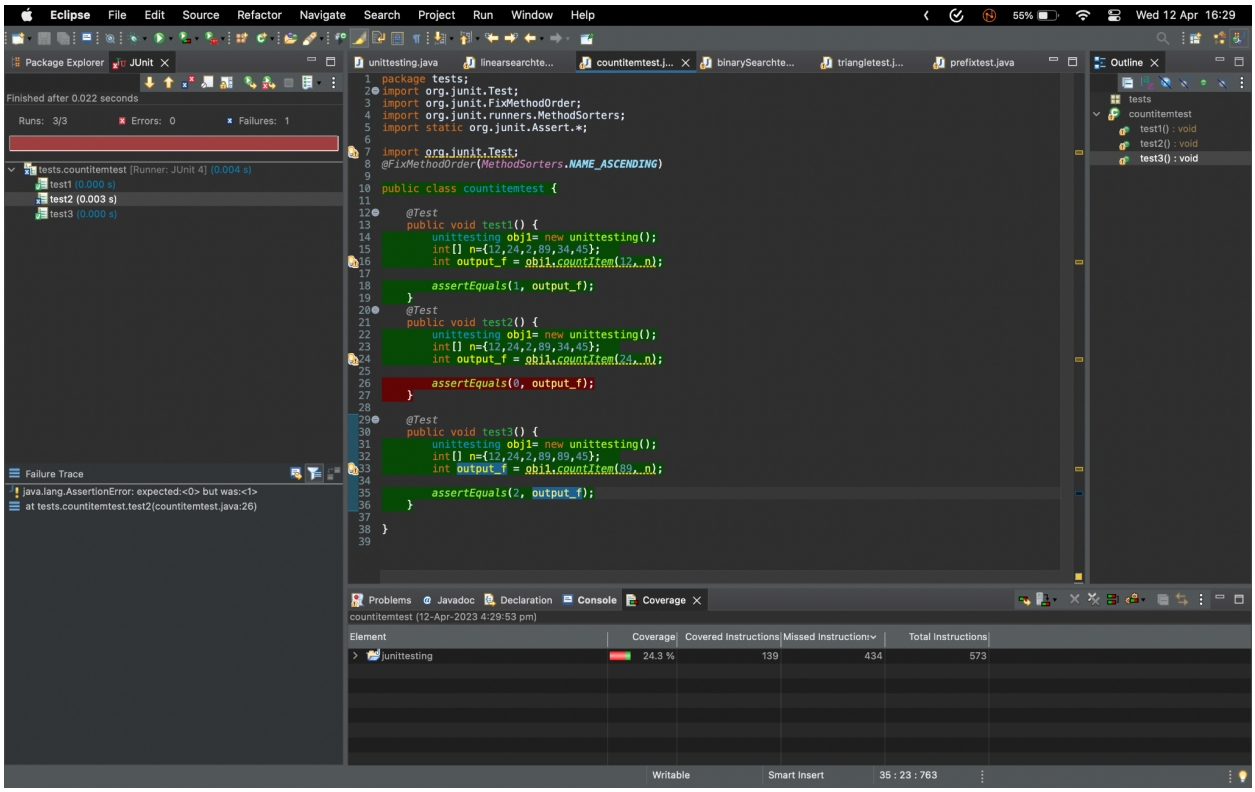
P1




```java
package tests;
import org.junit.Test;
import org.junit.FixMethodOrder;
import org.junit.runners.MethodSorters;
import static org.junit.Assert.*;
//import org.junit.Test;
@FixMethodOrder(MethodSorters.NAME_ASCENDING)
public class linearsearchtest {
	@Test
	public void test1() {
		unittesting obj1= new unittesting();
		int[] n={12,24,2,89,34,45};
		int output_f = obj1.linearSearch(12, n);

		assertEquals(0, output_f);
	}
	@Test
	public void test2() {
		unittesting obj1= new unittesting();
		int[] n={12,24,2,89,34,45};
		int output_f = obj1.linearSearch(24, n);
```

```
            assertEquals(1, output_f);
        }

        @Test
        public void test3() {
            unittesting obj1= new unittesting();
            int[] n={12,24,2,89,34,45};
            int output_f = obj1.linearSearch(45, n);

            assertEquals(10, output_f);
        }
}
```

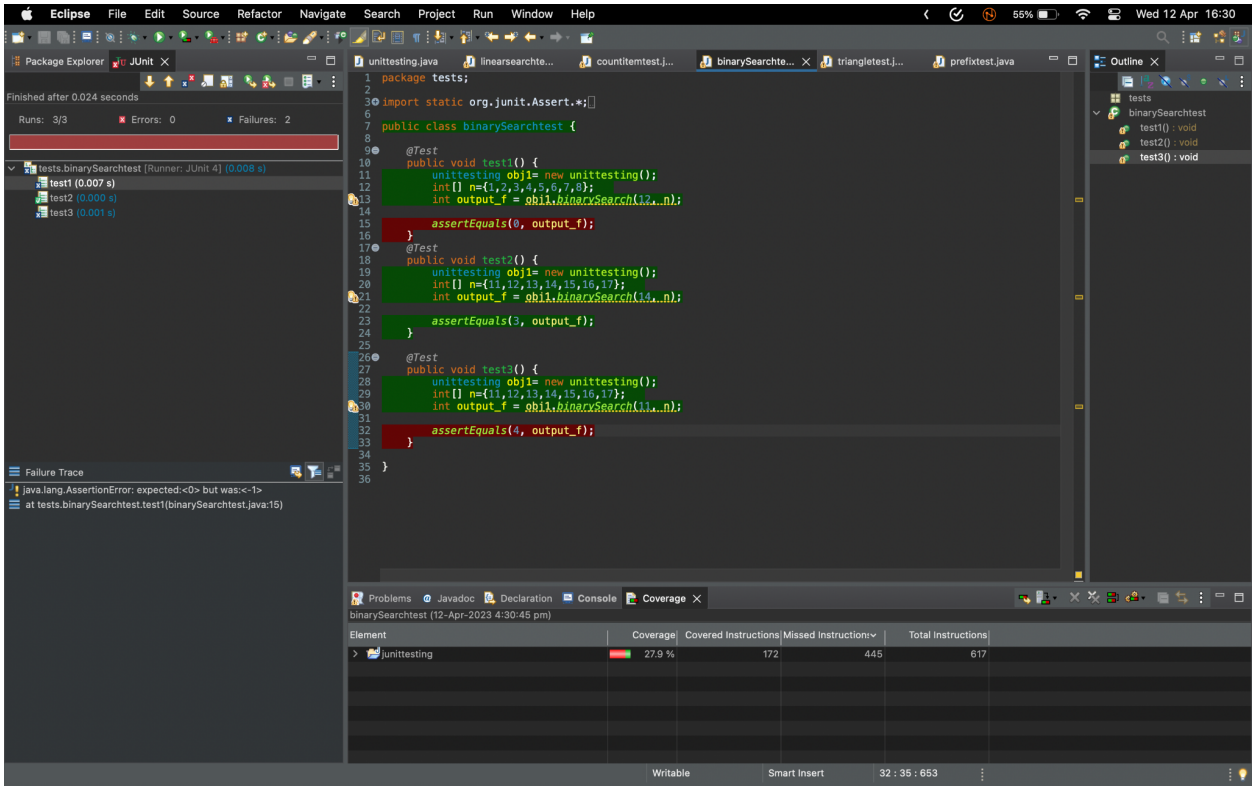
P2




```
package tests;
import org.junit.Test;
import org.junit.FixMethodOrder;
import org.junit.runners.MethodSorters;
import static org.junit.Assert.*;
import org.junit.Test;
```

```java
@FixMethodOrder(MethodSorters.NAME_ASCENDING)
public class countitemtest {
	@Test
	public void test1() {
		unittesting obj1= new unittesting();
		int[] n={12,24,2,89,34,45};
		int output_f = obj1.countItem(12, n);

		assertEquals(1, output_f);
	}
	@Test
	public void test2() {
		unittesting obj1= new unittesting();
		int[] n={12,24,2,89,34,45};
		int output_f = obj1.countItem(24, n);

		assertEquals(0, output_f);
	}

	@Test
	public void test3() {
		unittesting obj1= new unittesting();
		int[] n={12,24,2,89,89,45};
		int output_f = obj1.countItem(89, n);

		assertEquals(2, output_f);
	}
}
```

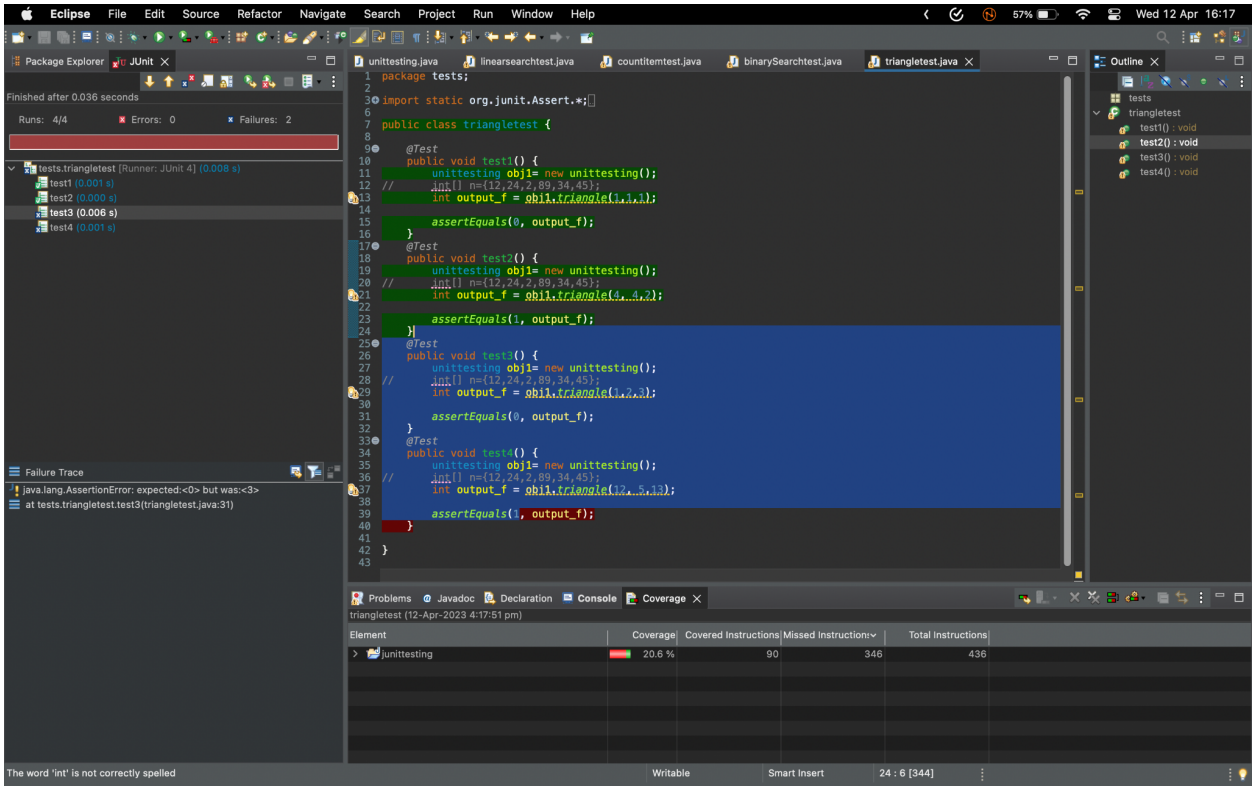
P3




```
package tests;

import static org.junit.Assert.*;

import org.junit.Test;

public class binarySearchtest {

	@Test
	public void test1() {
		unittesting obj1= new unittesting();
		int[] n={1,2,3,4,5,6,7,8};
		int output_f = obj1.binarySearch(12, n);

		assertEquals(0, output_f);
	}
	@Test
	public void test2() {
		unittesting obj1= new unittesting();
		int[] n={11,12,13,14,15,16,17};
		int output_f = obj1.binarySearch(14, n);

		assertEquals(3, output_f);
	}
```

```
	@Test
	public void test3() {
		unittesting obj1= new unittesting();
		int[] n={11,12,13,14,15,16,17};
		int output_f = obj1.binarySearch(11, n);

		assertEquals(4, output_f);
	}

}
```

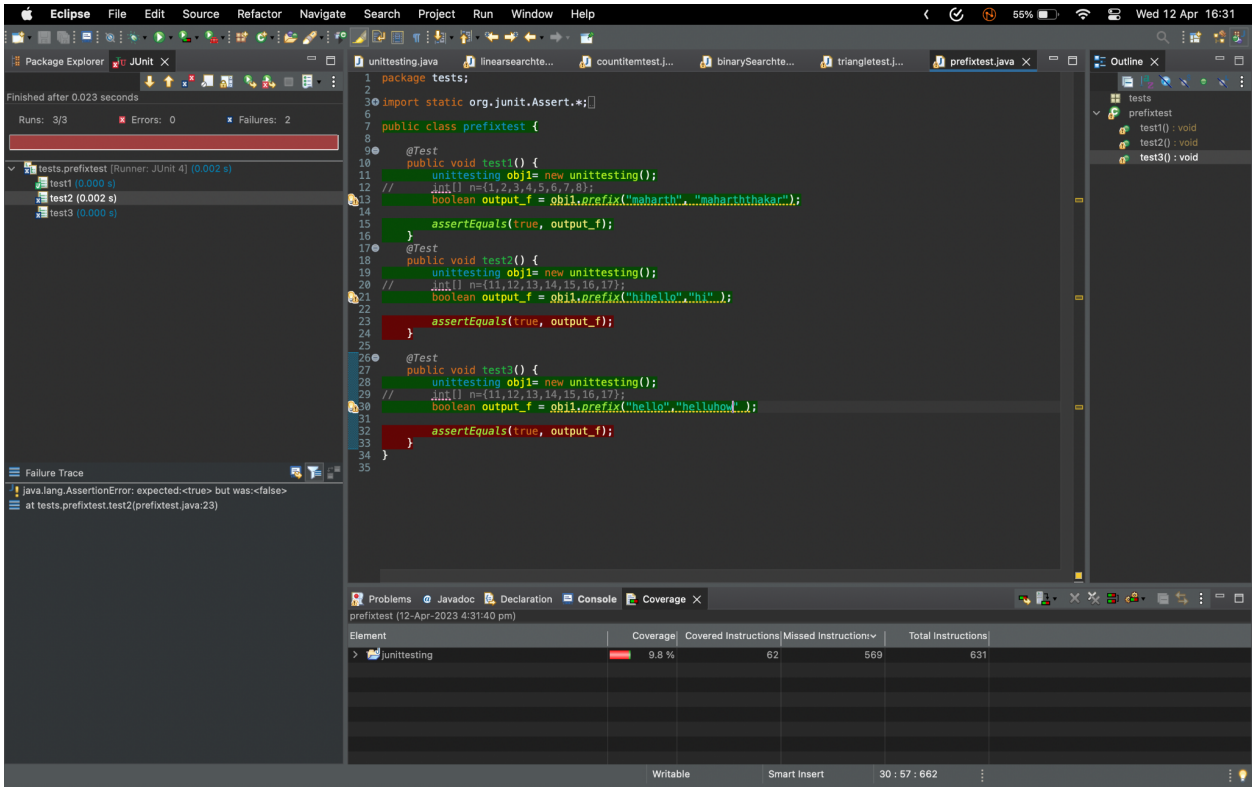
P4




```
package tests;

import static org.junit.Assert.*;

import org.junit.Test;

public class triangletest {

	@Test
	public void test1() {
		unittesting obj1= new unittesting();
//		int[] n={12,24,2,89,34,45};
		int output_f = obj1.triangle(1,1,1);

		assertEquals(0, output_f);
	}
	@Test
	public void test2() {
		unittesting obj1= new unittesting();
//		int[] n={12,24,2,89,34,45};
		int output_f = obj1.triangle(4, 4,2);

		assertEquals(1, output_f);
	}
```

```
	@Test
	public void test3() {
		unittesting obj1= new unittesting();
//		int[] n={12,24,2,89,34,45};
		int output_f = obj1.triangle(1,2,3);

		assertEquals(0, output_f);
	}
	@Test
	public void test4() {
		unittesting obj1= new unittesting();
//		int[] n={12,24,2,89,34,45};
		int output_f = obj1.triangle(12, 5,13);

		assertEquals(1, output_f);
	}

}
```

P5




```
package tests;
```

```
import static org.junit.Assert.*;

import org.junit.Test;

public class prefixtest {

	@Test
	public void test1() {
		unittesting obj1= new unittesting();
//		int[] n={1,2,3,4,5,6,7,8};
		boolean output_f = obj1.prefix("maharth", "maharththakar");

		assertEquals(true, output_f);
	}
	@Test
	public void test2() {
		unittesting obj1= new unittesting();
//		int[] n={11,12,13,14,15,16,17};
		boolean output_f = obj1.prefix("hihello","hi" );

		assertEquals(true, output_f);
	}

	@Test
	public void test3() {
		unittesting obj1= new unittesting();
//		int[] n={11,12,13,14,15,16,17};
		boolean output_f = obj1.prefix("hello","helluhow" );

		assertEquals(true, output_f);
	}
}
```

P6

a) Equivalence classes for the system are:

Class 1: Invalid inputs (negative or zero values)
Class 2: Non-triangle (sum of the two shorter sides is not greater than the longest side)
Class 3: Scalene triangle (no sides are equal)

Class 4: Isosceles triangle (two sides are equal)
Class 5: Equilateral triangle (all sides are equal)
Class 6: Right-angled triangle (satisfies the Pythagorean theorem)

b) Test cases to cover the identified equivalence classes:

Class 1: -1, 0
Class 2: 1, 2, 5
Class 3: 3, 4, 5
Class 4: 5, 5, 7
Class 5: 6, 6, 6
Class 6: 3, 4, 5
Test case 1 covers class 1, test case 2 covers class 2, test case 3 covers class 3, test case 4 covers class 4, test case 5 covers class 5, and test case 6 covers class 6.

c) Test cases to verify the boundary condition A + B > C for the scalene triangle:

2, 3, 6
3, 4, 8
Both test cases have two sides that are shorter than the third side, and should not form a triangle.

d) Test cases to verify the boundary condition A = C for the isosceles triangle:

2, 3, 3
5, 6, 5
Both test cases have two sides that are equal, and should form an isosceles triangle.

e) Test cases to verify the boundary condition A = B = C for the equilateral triangle:

5, 5, 5
9, 9, 9
Both test cases have all sides equal, and should form an equilateral triangle.

f) Test cases to verify the boundary condition A^2 + B^2 = C^2 for the right-angled triangle:

3, 4, 5
5, 12, 13
Both test cases satisfy the Pythagorean theorem and should form a right-angled triangle.

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

2, 2, 4
3, 6, 9
Both test cases have two sides that add up to the third side, and should not form a triangle.

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

0, 1, 2
-1, -2, -3
Both test cases have at least one non-positive value, which is an invalid input.