

LAB-7

Name: Tank Nandani Jagdishbhai
Student ID:202001201

Section - A

Design the equivalence class test cases

Equivalence classes

EC1 - date ≥ 1 and date ≤ 31
EC2 - date < 1
EC3 - date > 31
EC4 - month ≥ 1 and month ≤ 12
EC5 - month < 1
EC6 - month > 12
EC7 - year ≥ 1900 and year ≤ 2015
EC8 - year < 1900
EC9 - year > 2015

The dates should not be invalid - 31-02-2020 is an invalid date since month 2 does not have 31 days.

12 - 13 - 2005 is an example of a range date.

Equivalent test cases

EC1 - day - 3, month - 3, year - 2011
output - 2-03-2011

EC2 - day - 0, month - 3, year - 2012
output - Invalid date

EC3 - day - 41, month - 3, year - 2012
output - Invalid date

EC4 - day - 1, month - 1, year - 1980
output - 31/12/1979

EC5 - day - 20, month - -3, year - 2012
output - Invalid date

EC6 - day - 20, month - 15, year - 2013
output - Invalid date

EC8 - day - 5, month - 6, year - 1899
output - Invalid date

EC9 - day - 4, month - 3, year - 2021
output - Invalid date

Valid dates - calculate previous dates

1. 14-12-2011
output - 13-12-2011

2. 1-1-2014
output - 31-12-2013

Invalid dates

1. 31-4-2016
2. 14-45-1899

Out-of-range dates

1. 15-4-1899
2. 15-5-2022

Boundary Value Analysis

1. Earliest date - 1-1-1900
2. Last possible date - 31-12-2015
3. leap year - 29-2-2000
4. invalid leap year date - 29-2-2001

5. previous of earliest date - 31-12-1899
6. a day after latest date - 1-1-2016

PROGRAMS P1

The function linear search searches for a value v in an array of integers a . If v appears in the array a , then the function returns the first index i , such that $a[i] == v$; otherwise, -1 is returned.

```
public class lab7_1 {  
    public static int linearSearch(int v, int[] a) {  
        int i = 0;  
        while (i < a.length) {  
            if (a[i] == v) {  
                return i;  
            }  
            i++;  
        }  
        return -1;  
    }  
}
```

Equivalence Partitioning

V is present in an	Index v
V is not present	-1

Boundary Value Analysis

Empty array a	-1
V is present at the first index of a	0
V is present at the last index of a	-1

length of a	
V is not present in a	-1

TEST CASES WHICH WE TESTED

1. $v = 3$; $a[] = \{1,2,3\}$; expected = 2

Test case passed!

The screenshot shows the Eclipse IDE interface. The main editor displays the source code for `linearSearch.java`. The code includes a package declaration, imports, and a class with a test method. The left sidebar shows the Package Explorer with the `tests` package selected. The bottom status bar indicates the test run was successful.

```
//202001201
1 package tests;
2
3
4 import static org.junit.jupiter.api.Assertions.*;
5
6 class linearSearch {
7
8     @Test
9     void test() {
10         Test1 obj1=new Test1();
11         int v=3;
12         int a[]={1,2,3};
13         int output=obj1.linearSearch(v,a);
14         int expected=2;
15         assertEquals(expected,output);
16     }
17 }
18
19
20
21
```

JUnit Test Results:

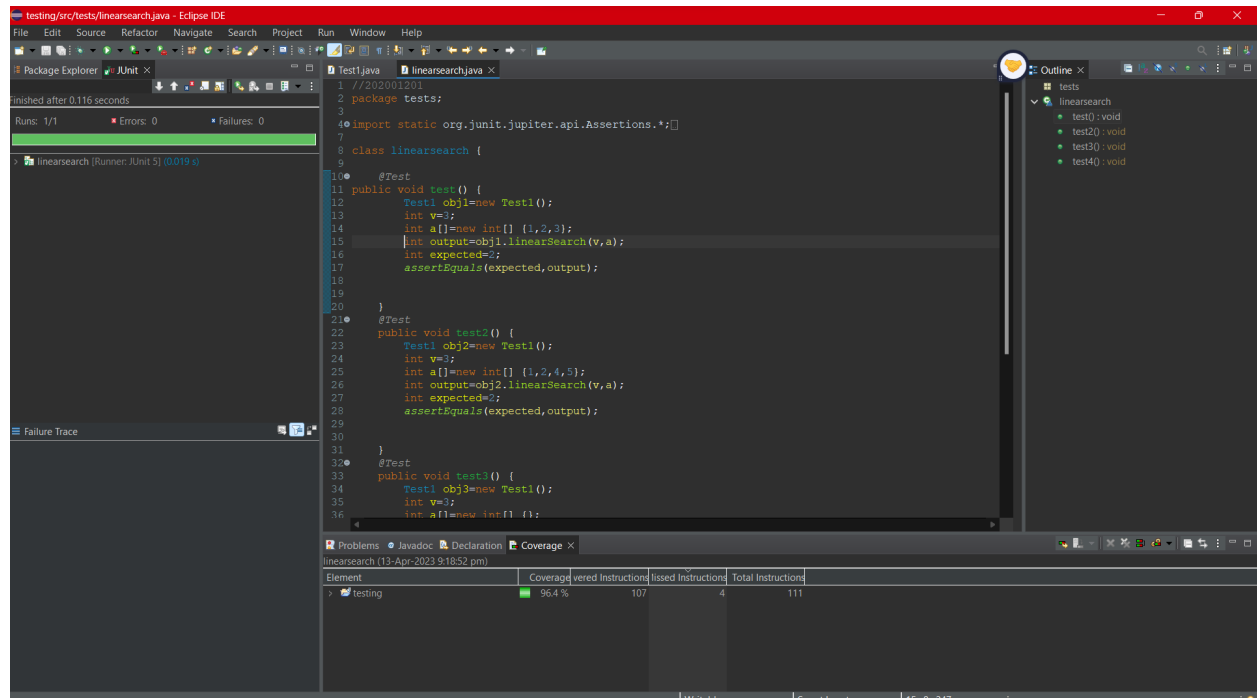
Test Case	Result	Time (s)
linearSearch (Runner: JUnit 5)	Passed	0.0000

Problems: 0 items

Description	Resource	Path	Location	Type
-------------	----------	------	----------	------

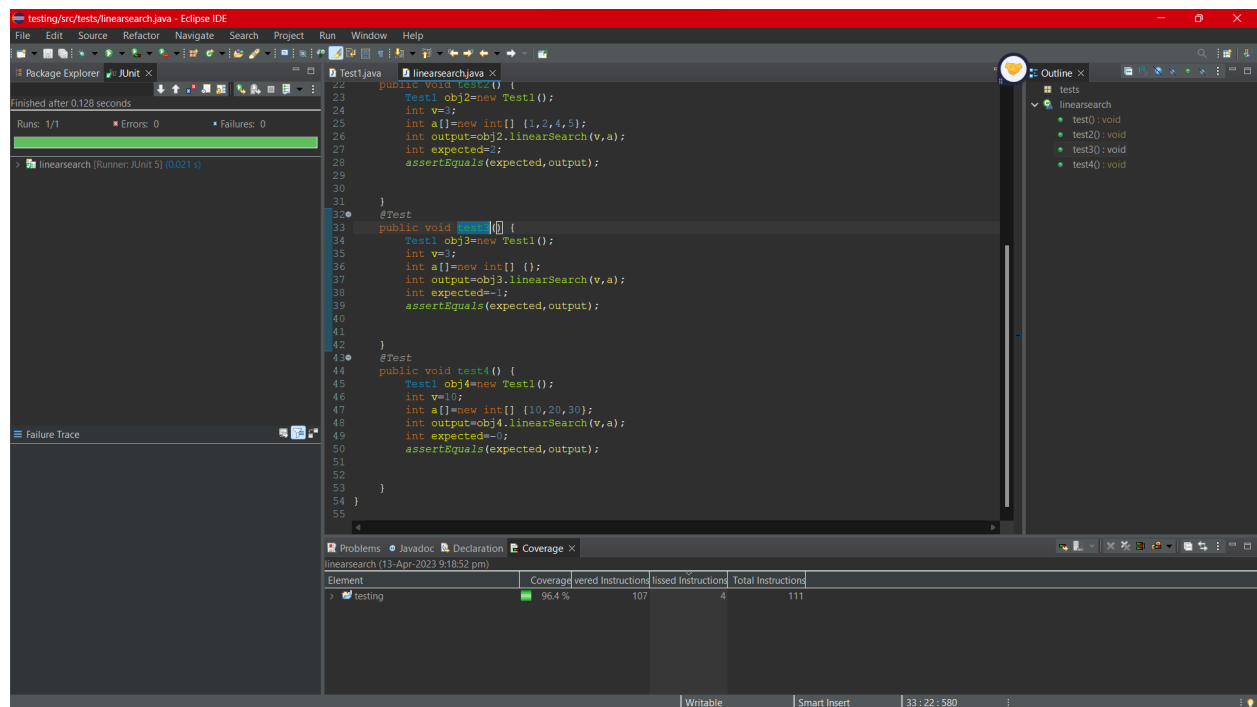
3. $v = 3$; $a[] = \{\}$; expected = -1;

Test case Passed!



4. v = 10; a[] = {10,20,30}, expected = 0;

Test case Passed!



PROGRAM P2

The function counter returns the number of times a value v appears in an array of integers a.

```
public class lab7_2 {  
    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);  
    }  
}
```

Equivalence Partitioning

V is present in a	number of times v appears in
V is not present in a	0

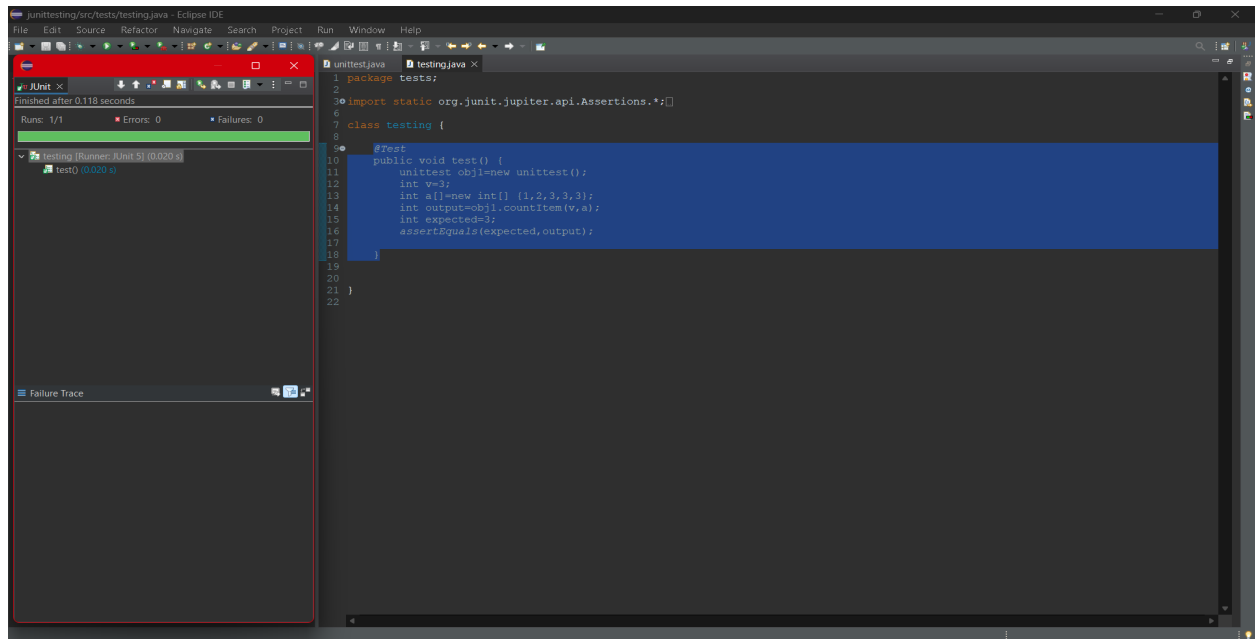
Boundary Value Analysis

Empty array a	0
V is present once in a	1
V is present multiple times in a	Number of times V appears in a
V is present at the first index of a	1
V is present at the last index of a	1
V is not present in a	0

TEST CASES

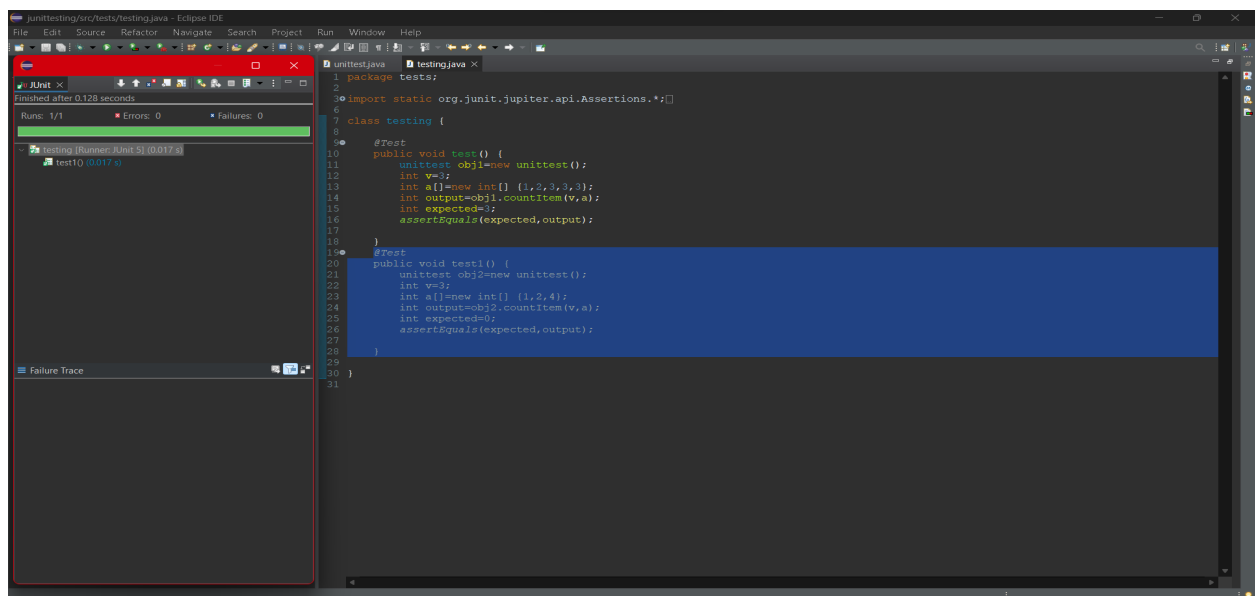
1. $v = 3$; $a[] = \{1,2,3,3,3\}$; expected = 3

Test case passed!



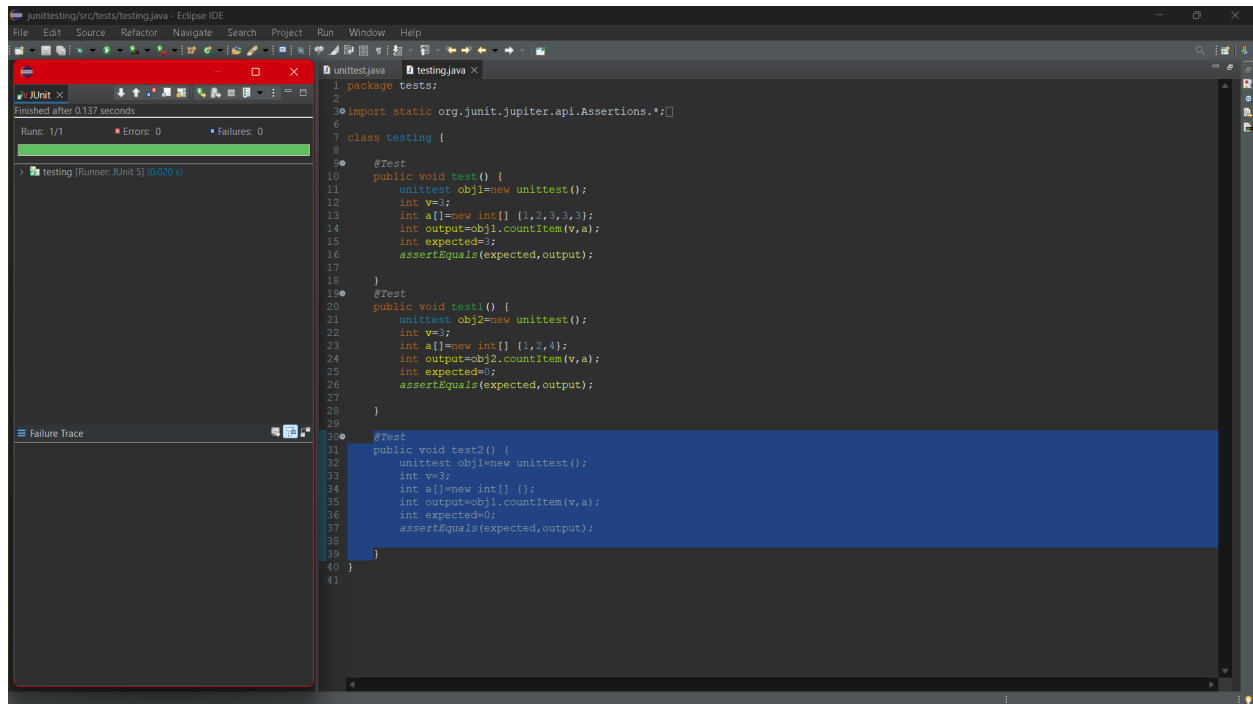
2. $v = 3$; $a[] = \{1,2,4\}$; expected = 0

Test case passed!



3. `v = 3; a[] = {};` expected = 0

Test case passed!

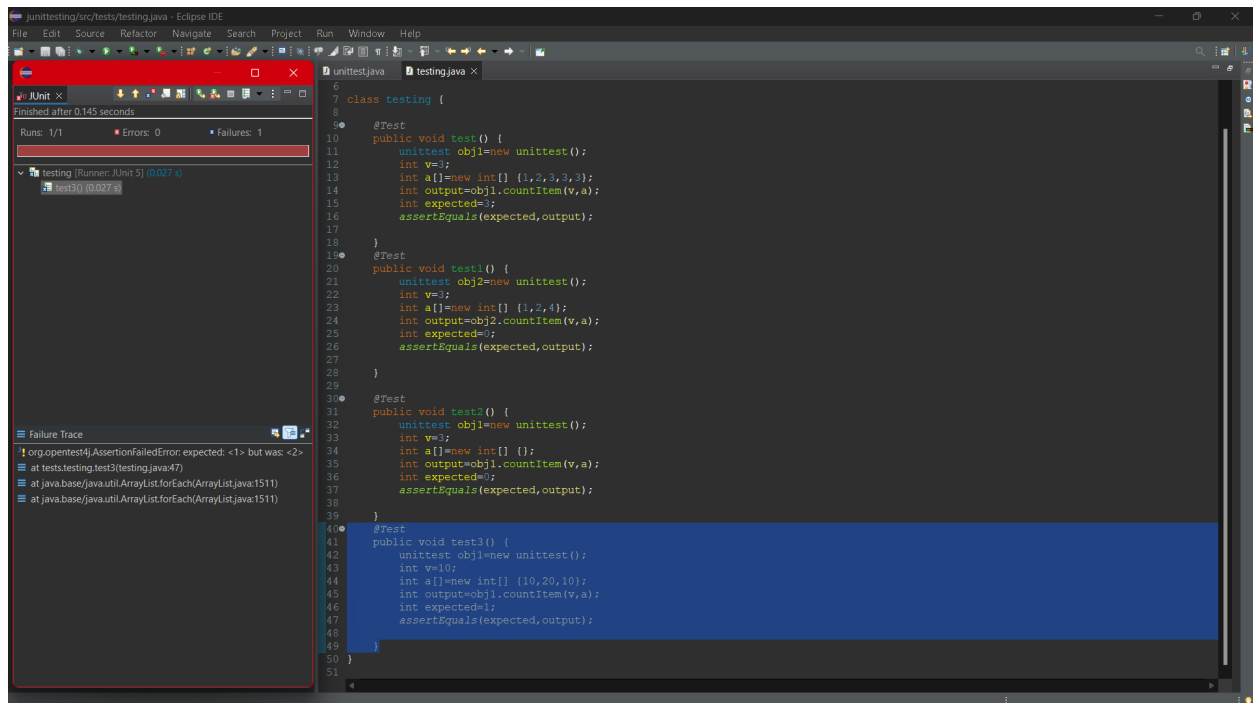


The screenshot shows the Eclipse IDE with a JUnit test run. The left sidebar displays the 'JUnit' window with a progress bar indicating 'Runs: 1/1', 'Errors: 0', and 'Failures: 0'. The 'testing' test is listed as 'testing [Runner: JUnit 5] (0.020 s)'. The main editor shows the source code of the 'testing' class, which contains three test methods: 'test()', 'test1()', and 'test2()'. Each method creates a 'unittest' object, sets 'v' to 3, initializes an array 'a' with [1, 2, 3, 3, 3], and calls 'countItem(v, a)' to get 'output'. The 'test()' method expects 3, 'test1()' expects 0, and 'test2()' expects 0. All three tests are marked as passed.

```
1 package tests;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5 class testing {
6
7     @Test
8     public void test() {
9         unittest obj1=new unittest();
10        int v=3;
11        int a[]=new int[] {1,2,3,3,3};
12        int output=obj1.countItem(v,a);
13        int expected=3;
14        assertEquals(expected,output);
15    }
16
17     @Test
18     public void test1() {
19         unittest obj2=new unittest();
20        int v=3;
21        int a[]=new int[] {1,2,4};
22        int output=obj2.countItem(v,a);
23        int expected=0;
24        assertEquals(expected,output);
25    }
26
27     @Test
28     public void test2() {
29         unittest obj1=new unittest();
30        int v=3;
31        int a[]=new int[] {};
32        int output=obj1.countItem(v,a);
33        int expected=0;
34        assertEquals(expected,output);
35    }
36
37 }
38
39
40
41
```

4. `v = 10; a[] = {10, 20, 10};` expected = 1

test case failed!



The screenshot shows the Eclipse IDE with a JUnit test run. The left sidebar displays the 'JUnit' window with a progress bar indicating 'Runs: 1/1', 'Errors: 0', and 'Failures: 1'. The 'testing' test is listed as 'testing [Runner: JUnit 5] (0.027 s)'. The 'Failure Trace' window shows the following error: 'org.opentest4j.AssertionFailedError: expected: <1> but was: <2>'. The main editor shows the source code of the 'testing' class, which contains three test methods: 'test()', 'test1()', and 'test2()'. Each method creates a 'unittest' object, sets 'v' to 10, initializes an array 'a' with [10, 20, 10], and calls 'countItem(v, a)' to get 'output'. The 'test()' method expects 1, 'test1()' expects 0, and 'test2()' expects 0. The 'test()' method is marked as failed.

```
1 package tests;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5 class testing {
6
7     @Test
8     public void test() {
9         unittest obj1=new unittest();
10        int v=10;
11        int a[]=new int[] {10,20,10};
12        int output=obj1.countItem(v,a);
13        int expected=1;
14        assertEquals(expected,output);
15    }
16
17     @Test
18     public void test1() {
19         unittest obj2=new unittest();
20        int v=10;
21        int a[]=new int[] {1,2,4};
22        int output=obj2.countItem(v,a);
23        int expected=0;
24        assertEquals(expected,output);
25    }
26
27     @Test
28     public void test2() {
29         unittest obj1=new unittest();
30        int v=10;
31        int a[]=new int[] {};
32        int output=obj1.countItem(v,a);
33        int expected=0;
34        assertEquals(expected,output);
35    }
36
37 }
38
39
40
41
42
43
44
45
46
47
48
49
50
51
```


PROGRAM P3

The function `binarySearch` searches for a value `v` in an ordered array of integers `a`. If `v` appears in the array `a`, then the function returns an index `i`, such that `a[i] == v`; otherwise, `-1` is returned. Assumption: the elements in the array `a` are sorted in non-decreasing order.

```
public class lab7_3 {  
    public static int binarySearch(int v, int a[])  
    {  
        int lo, mid, hi;  
        lo = 0;  
        hi = a.length-1;  
        while (lo <= hi)  
        {  
            mid = (lo+hi)/2;  
            if (v == a[mid])  
                return (mid);  
            else if (v < a[mid])  
                hi = mid-1;  
            else  
                lo = mid+1;  
        }  
        return(-1);  
    }  
}
```

Equivalence Partitioning

V is present in an	Index of v
V is not present in a	-1

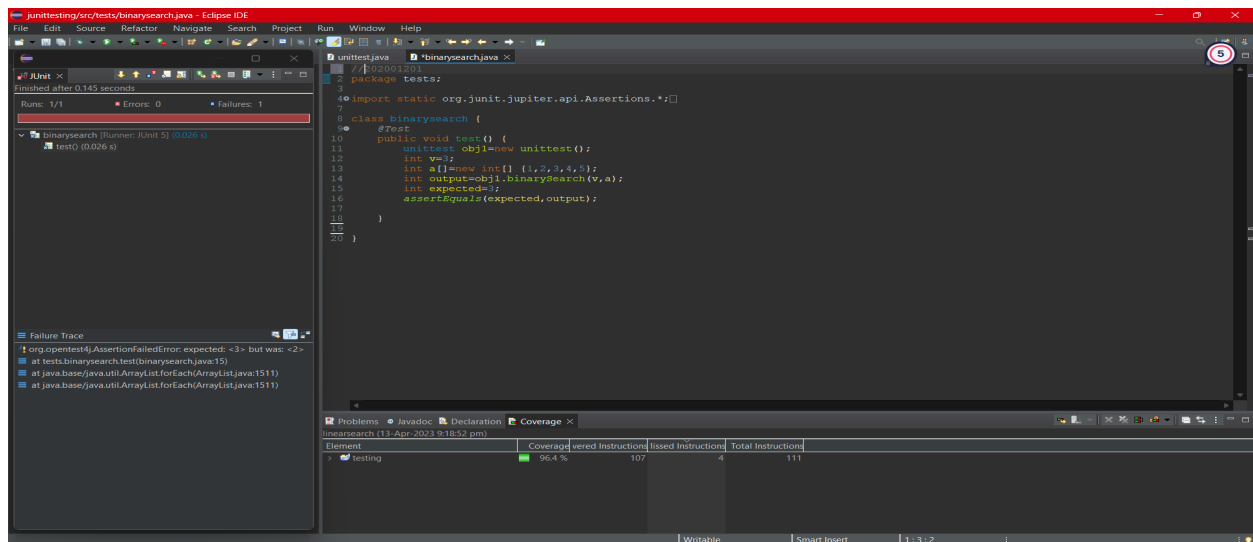
Boundary Value Analysis

Empty array a	-1
V is present at the first index of a	0
V is present at the last index of a length of a	-1
V is not present in a	-1

TEST CASES

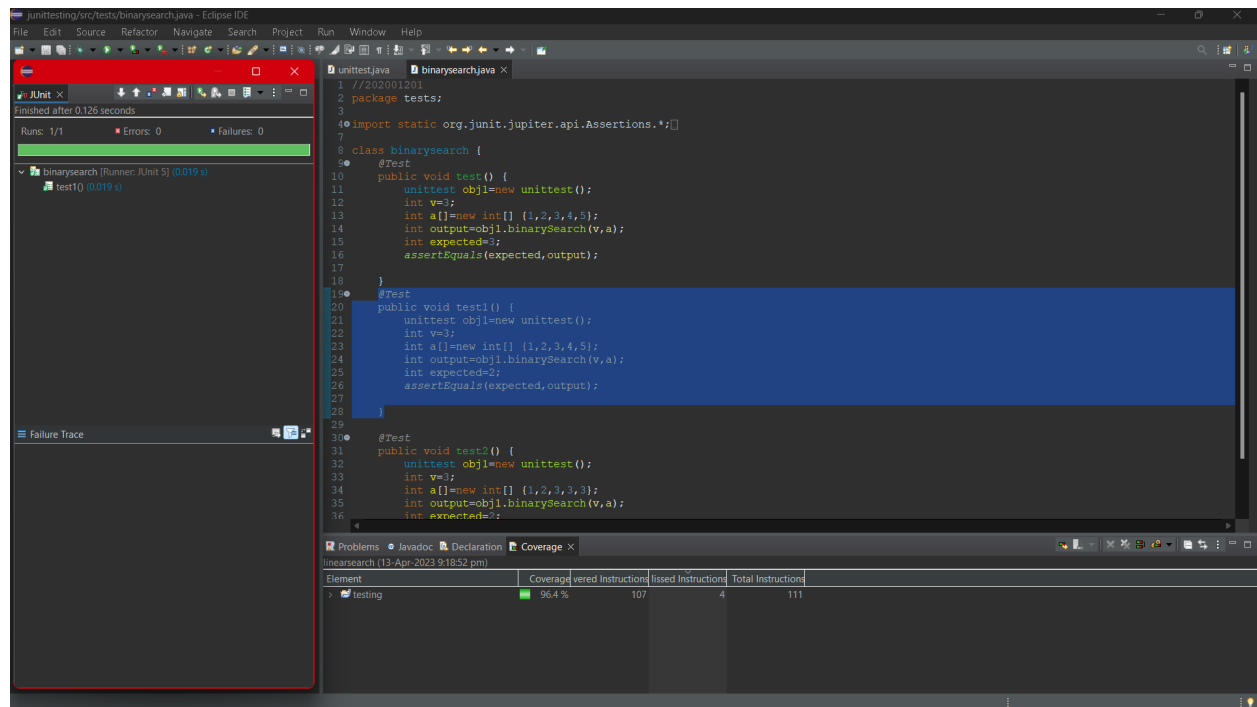
1. $v = 3$; $a[] = \{1,2,3,4,5\}$; expected = 3

Test case failed!



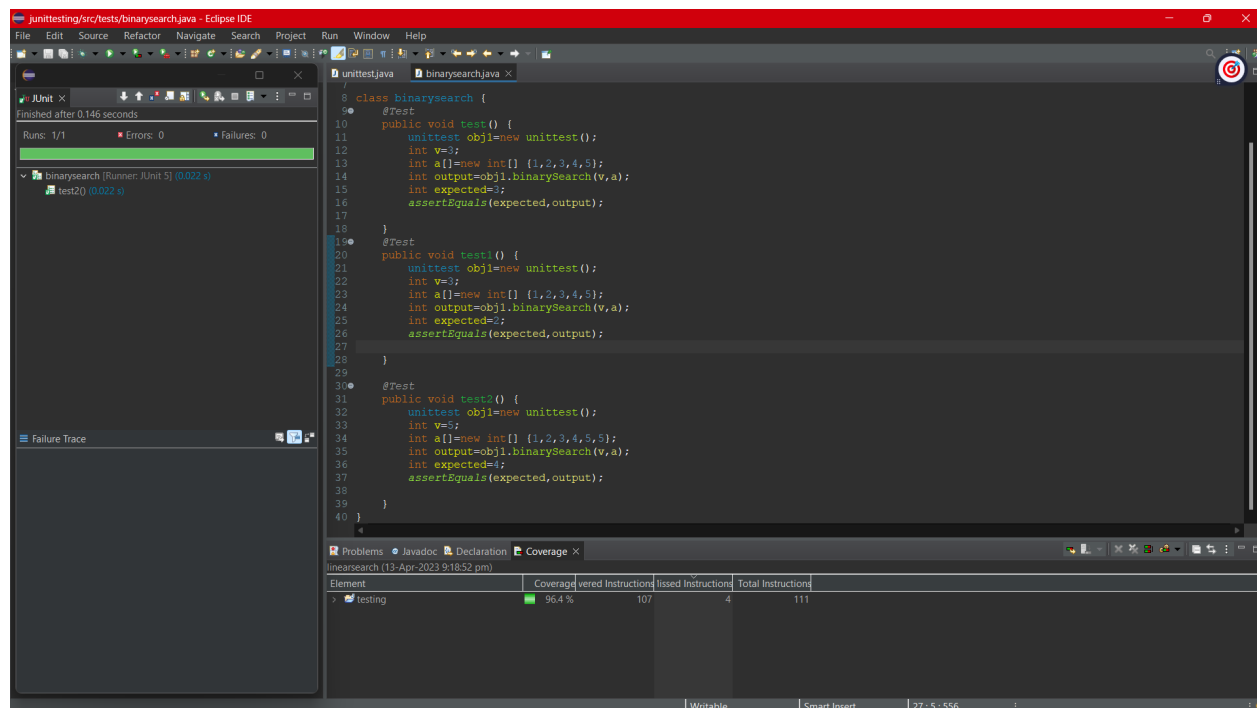
2. $v = 3$; $a[] = \{1,2,3,4,5\}$; expected = 2

Test case passed!



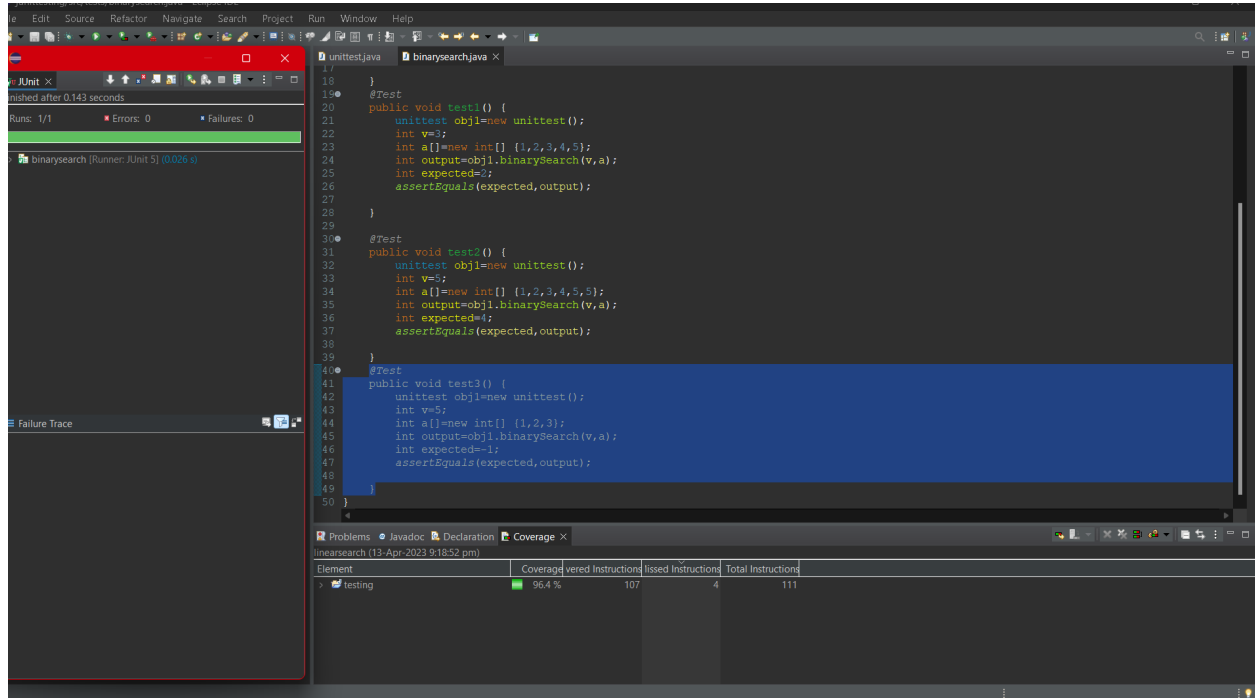
3. v = 5; a[] = {1,2,3,4,5,5}; expected = 4

Test case passed!



4. $v = 5$; $a[] = \{1,2,3\}$; expected = -1

Test case passed!



```
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
}
```

JUnit X
Finished after 0.143 seconds
Runs: 1/1
Errors: 0
Failures: 0
Binarysearch [Runner: JUnit 5] (0.026 s)

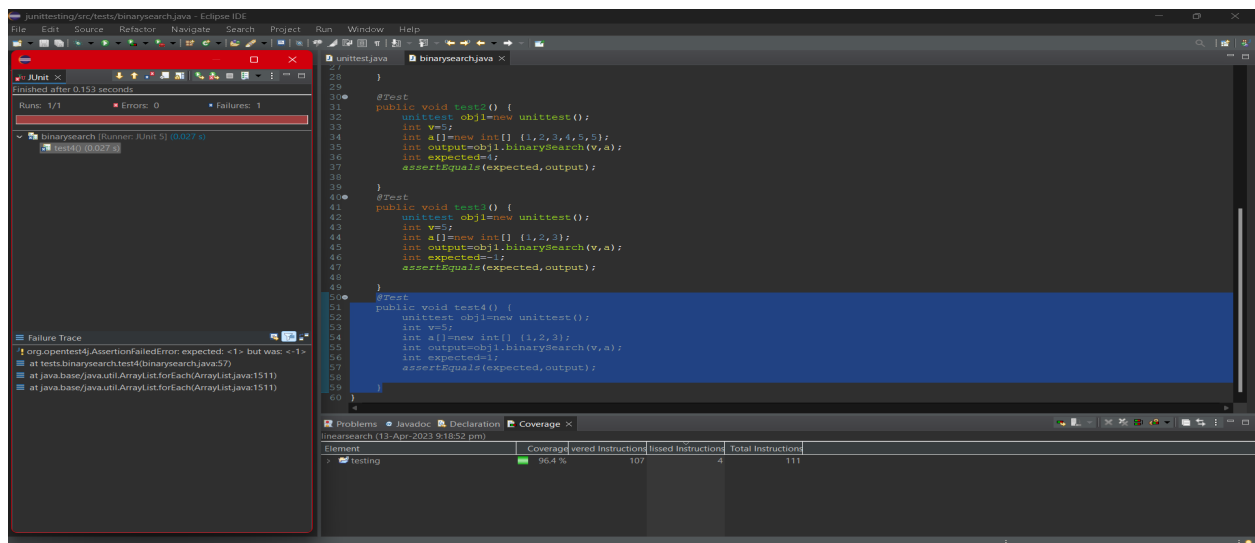
Failure Trace

Problems Javadoc Declaration Coverage X
Binarysearch (13-Apr-2023 9:18:52 pm)

Element	Coverage	Vered Instructions	Issed Instructions	Total Instructions
testing	96.4 %	107	4	111

5. $v = 5$; $a[] = \{1,2,3\}$; expected = 1

Test case failed!



```
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
}
```

JUnit X
Finished after 0.153 seconds
Runs: 1/1
Errors: 0
Failures: 1
Binarysearch [Runner: JUnit 5] (0.027 s)
test4() (0.027 s)

Failure Trace

org.opentest4.AssertionFailedError: expected: <1> but was: <-1>
at tests.binarysearch.test4(binarysearch.java:57)
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)

Problems Javadoc Declaration Coverage X
Binarysearch (13-Apr-2023 9:18:52 pm)

Element	Coverage	Vered Instructions	Issed Instructions	Total Instructions
testing	96.4 %	107	4	111

PROGRAM P4

- The following problem has been adapted from The Art of Software Testing, by G. Myers (1979). The function triangle takes three integer parameters that are interpreted as the lengths of the sides of a triangle. It returns whether the triangle is equilateral (three lengths equal), isosceles (two lengths equal), scalene (no lengths equal), or invalid (impossible lengths).

```
public class lab7_4 {  
    final int EQUILATERAL = 0;  
    final int ISOSCELES = 1;  
    final int SCALENE = 2;  
    final int INVALID = 3;  
    public 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);  
    }  
}
```

Equivalence Partitioning

Invalid triangle($a+b \leq c$)	INVALID
Valid equilateral triangle($a=b=c$)	EQUILATERAL
Valid isosceles triangle($a=b < c$)	ISOSCELES
Valid scalene triangle($a < b < c$)	SCALENE

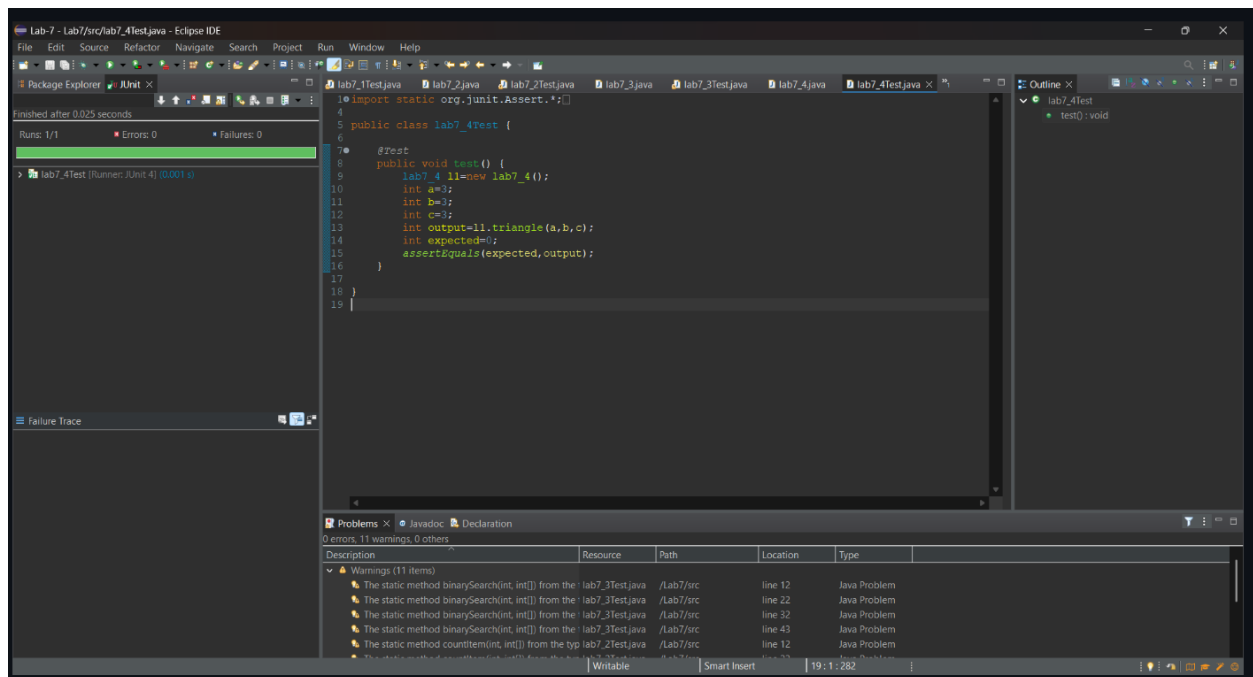
Boundary Value Analysis

Invalid triangle ($a+b \leq c$)	INVALID
Invalid triangle ($a+c \leq b$)	INVALID
Invalid triangle ($b+c \leq a$)	INVALID
Valid equilateral triangle ($a=b=c$)	EQUILATRAL
Valid isosceles triangle ($a=b < c$)	ISOSCELES
Valid isosceles triangle ($a=c < b$)	ISOSCELES
Valid isosceles triangle ($b=c < a$)	ISOSCELES
Valid scalene triangle ($a < b < c$)	SCALENE

TEST CASES

1. $a = 3$; $b=3$; $c = 3$; expected = 0 (equilateral)

Test case passed!



2. a = 1, b = 2, c = 3; expected = 0

Test case failed!

Lab-7 - Lab7/src/lab7_4Test.java - Eclipse IDE

Package Explorer: lab7_1Test.java, lab7_2.java, lab7_2Test.java, lab7_3.java, lab7_3Test.java, lab7_4.java, lab7_4Test.java

lab7_4Test (Runner: JUnit 4) (0.004 s)

- test (0.000 s)
- test1 (0.004 s)

Failure Trace

```
java.lang.AssertionError: expected<0> but was<3>
    at lab7_4Test.test1(lab7_4Test.java:26)
```

```
1 import static org.junit.Assert.*;
2
3 public class lab7_4Test {
4
5     @Test
6     public void test() {
7         lab7_4 ll=new lab7_4();
8         int a=3;
9         int b=3;
10        int c=3;
11        int output=ll.triangle(a,b,c);
12        int expected=0;
13        assertEquals(expected,output);
14    }
15
16    @Test
17    public void test1() {
18        lab7_4 ll=new lab7_4();
19        int a=1;
20        int b=2;
21        int c=3;
22        int output=ll.triangle(a,b,c);
23        int expected=0;
24        assertEquals(expected,output);
25    }
26
27 }
28
29
30
```

Problems: 0 errors, 11 warnings, 0 others

Description	Resource	Path	Location	Type
The static method binarySearch(int, int[]) from the type lab7_3Test.java	/Lab7/src	line 12	Java Problem	
The static method binarySearch(int, int[]) from the type lab7_3Test.java	/Lab7/src	line 22	Java Problem	
The static method binarySearch(int, int[]) from the type lab7_3Test.java	/Lab7/src	line 32	Java Problem	
The static method binarySearch(int, int[]) from the type lab7_3Test.java	/Lab7/src	line 43	Java Problem	
The static method countItem(int, int[]) from the type lab7_2Test.java	/Lab7/src	line 12	Java Problem	

3. a = -1, b = 2, c = 3, expected = 2

Test case failed!

Lab-7 - Lab7/src/lab7_4Test.java - Eclipse IDE

Package Explorer: lab7_1Test.java, lab7_2.java, lab7_2Test.java, lab7_3.java, lab7_3Test.java, lab7_4.java, lab7_4Test.java

lab7_4Test (Runner: JUnit 4) (0.003 s)

- test (0.000 s)
- test1 (0.000 s)
- test2 (0.003 s)

Failure Trace

```
java.lang.AssertionError: expected<2> but was<3>
    at lab7_4Test.test2(lab7_4Test.java:37)
```

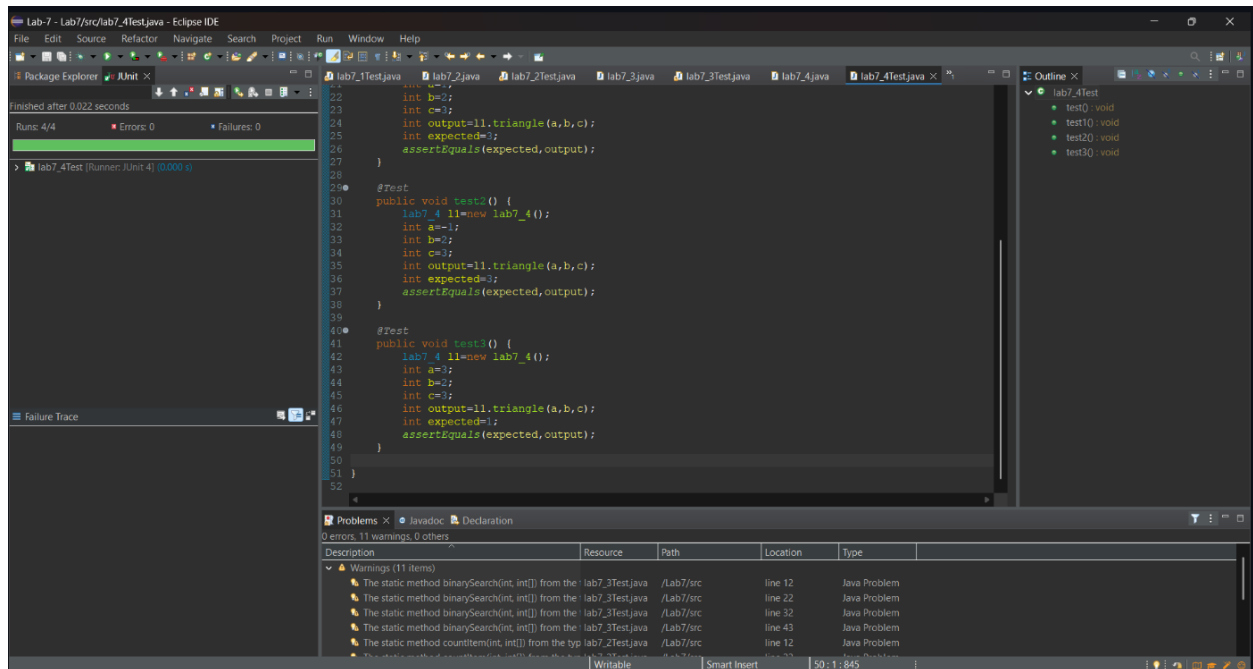
```
11 int b=3;
12 int c=3;
13 int output=ll.triangle(a,b,c);
14 int expected=0;
15 assertEquals(expected,output);
16
17
18 @Test
19 public void test1() {
20     lab7_4 ll=new lab7_4();
21     int a=1;
22     int b=2;
23     int c=3;
24     int output=ll.triangle(a,b,c);
25     int expected=3;
26     assertEquals(expected,output);
27 }
28
29 @Test
30 public void test2() {
31     lab7_4 ll=new lab7_4();
32     int a=-1;
33     int b=2;
34     int c=3;
35     int output=ll.triangle(a,b,c);
36     int expected=2;
37     assertEquals(expected,output);
38 }
39
40
41
```

Problems: 0 errors, 11 warnings, 0 others

Description	Resource	Path	Location	Type
The static method binarySearch(int, int[]) from the type lab7_3Test.java	/Lab7/src	line 12	Java Problem	
The static method binarySearch(int, int[]) from the type lab7_3Test.java	/Lab7/src	line 22	Java Problem	
The static method binarySearch(int, int[]) from the type lab7_3Test.java	/Lab7/src	line 32	Java Problem	
The static method binarySearch(int, int[]) from the type lab7_3Test.java	/Lab7/src	line 43	Java Problem	
The static method countItem(int, int[]) from the type lab7_2Test.java	/Lab7/src	line 12	Java Problem	

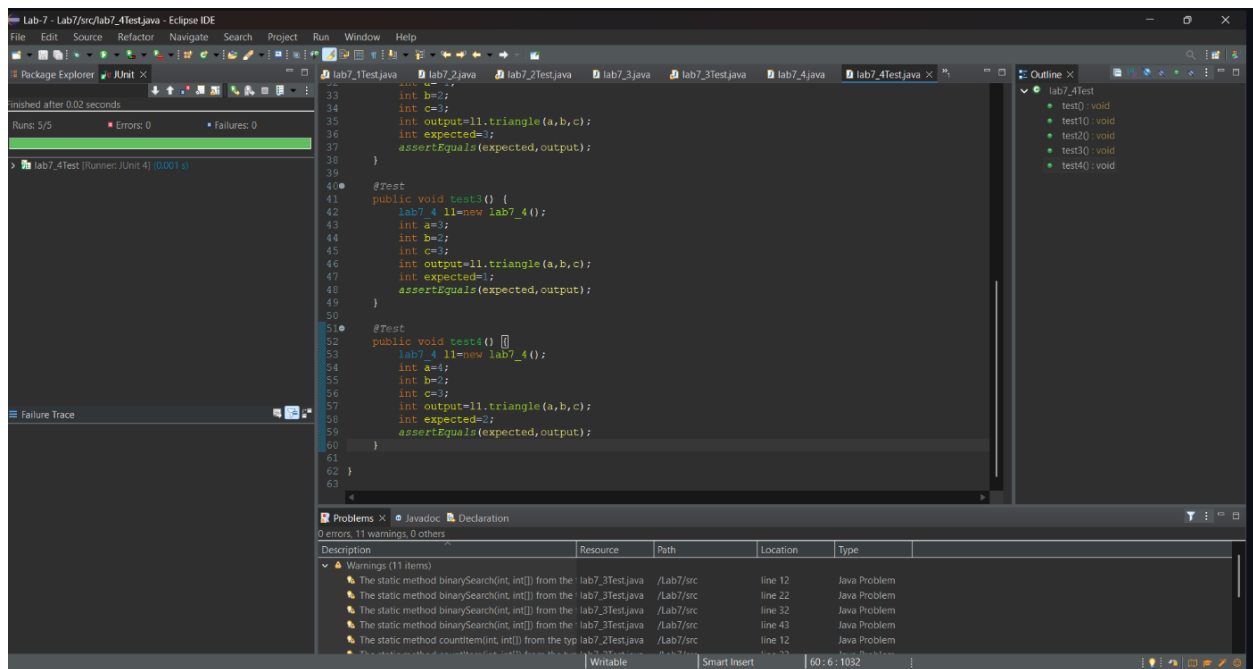
4. a = 3, b = 2, c = 3, expected = 1

Test case passed!



4. a = 4, b = 2, c = 3, expected = 2

Test case passed!



PROGRAM P5

- The function `prefix (String s1, String s2)` returns whether or not the string `s1` is a prefix of string `s2` (you may assume that neither `s1` nor `s2` is null).

```
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;
}
```

Equivalence Partitioning

Empty string s1 and s2 True	TRUE
Empty string s1 and non-empty s2	TRUE
Non-empty s1 is a prefix of non-empty	TRUE
Non-empty s1 is not a prefix of s2	FALSE
Non-empty s1 is longer than s2	FALSE

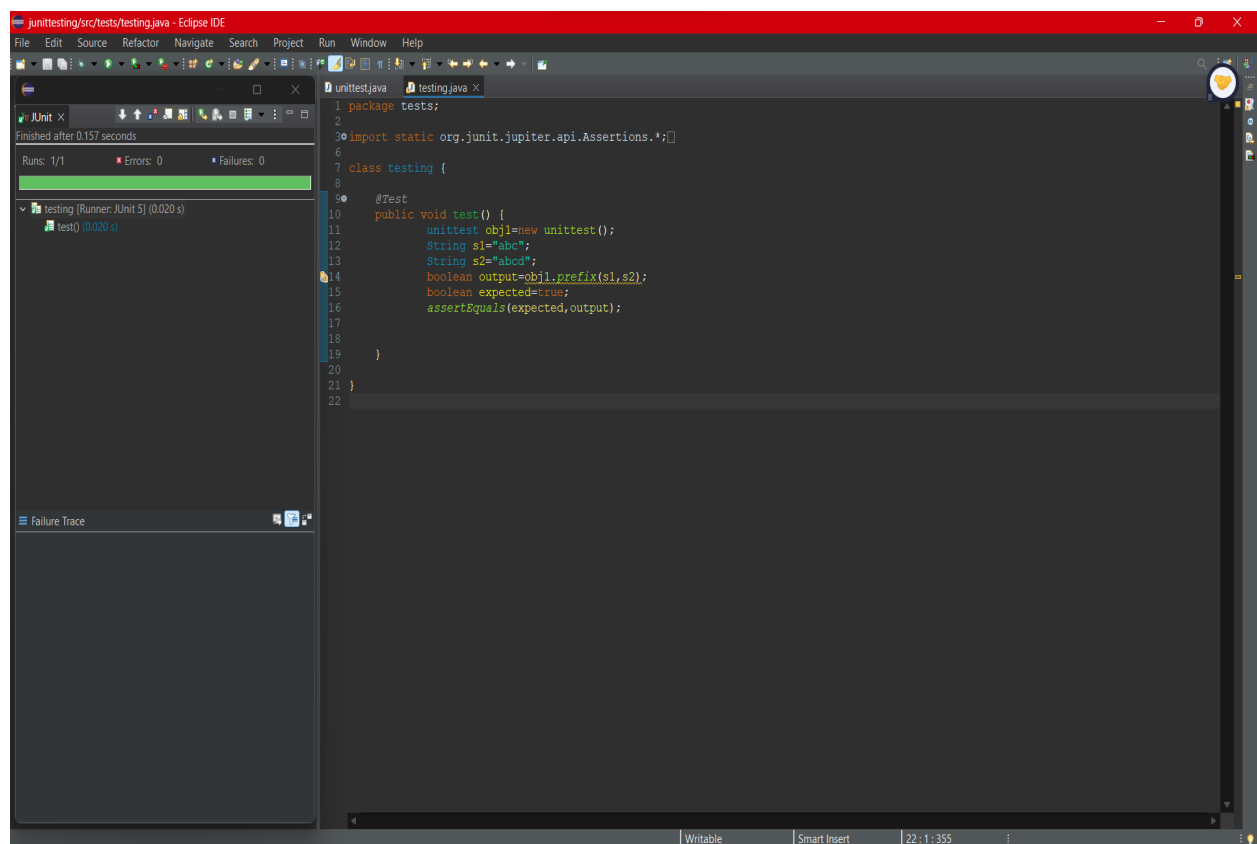
Boundary value analysis

Empty string s1 and s2	True
Empty string s1 and nonempty s2	True
S1 prefix of s2	True
S1 longer than s2	false

TEST CASES

1. String s1 = "abc", s2 = "abcd"; expected = true

Test case passed!



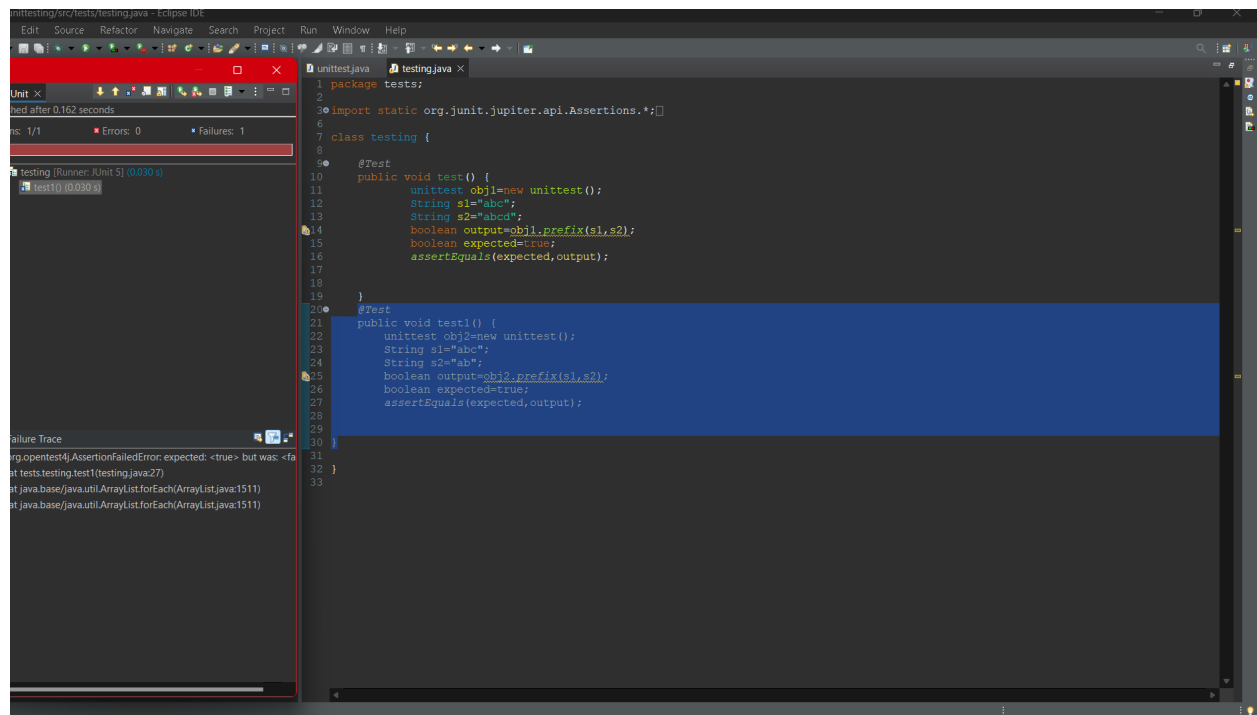
The screenshot shows the Eclipse IDE interface. The main editor displays the source code for a Java test class named `testing.java`. The code is as follows:

```
1 package tests;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5
6
7 class testing {
8
9     @Test
10    public void test() {
11        unittest obj=new unittest();
12        String s1="abc";
13        String s2="abcd";
14        boolean output=obj.prefix(s1,s2);
15        boolean expected=true;
16        assertEquals(expected,output);
17    }
18 }
19
20
21
22
```

The left sidebar shows the JUnit test runner. It indicates that the test was finished after 0.157 seconds, with 1/1 runs, 0 errors, and 0 failures. A green progress bar is visible. Below this, a tree view shows the test results for the `testing` class, with a sub-entry for the `test` method, which passed in 0.020 seconds. At the bottom of the sidebar, there is a 'Failure Trace' section, which is currently empty.

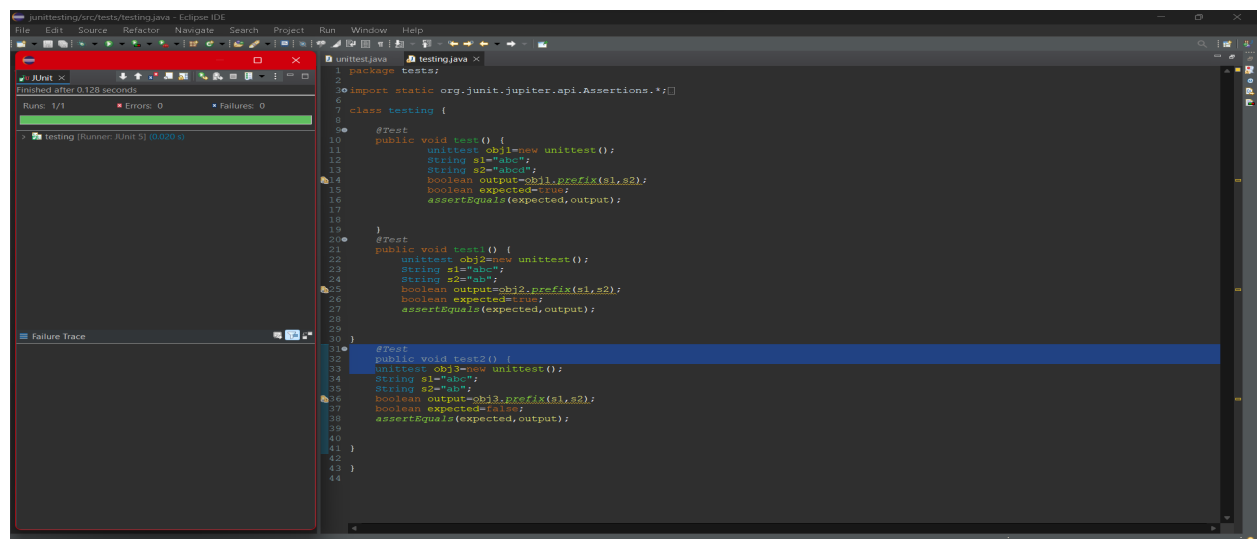
2. String s1 = "abc", s2 = "ab"; expected = true

Test case failed!



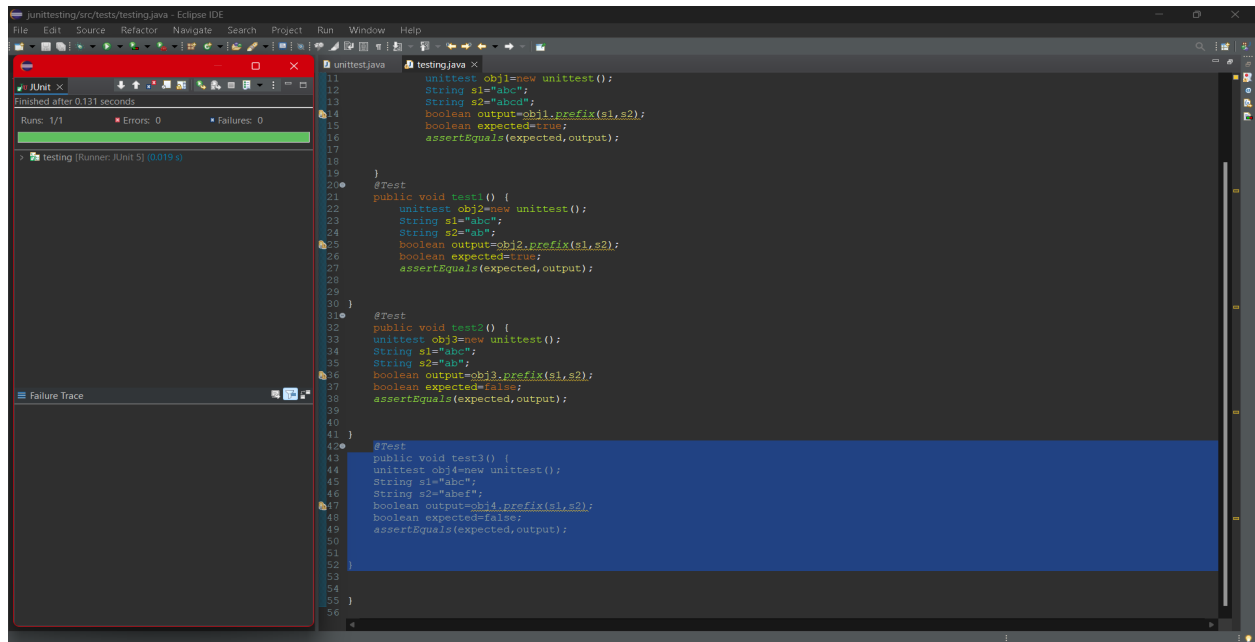
3. String s1 = "abc", s2 = "ab"; expected = false

Test case passed!



4. String s1 = "abc", s2 = "abef"; expected = false

Test case passed!

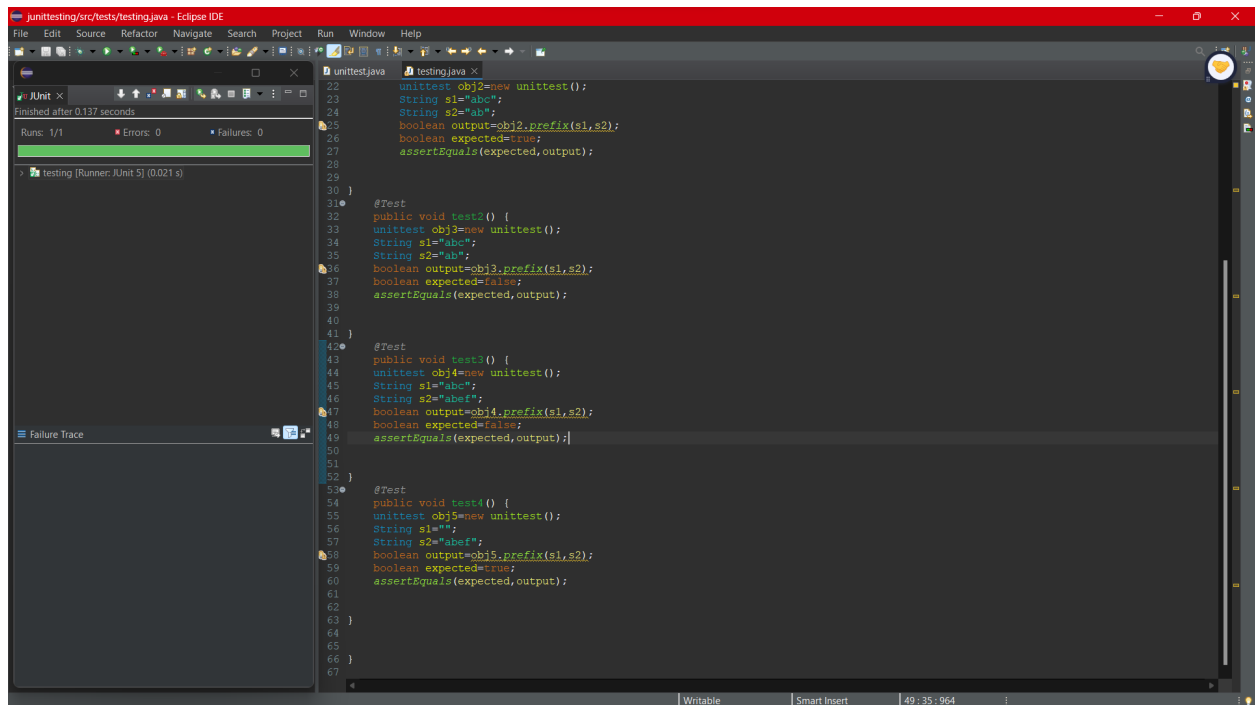


The screenshot shows the Eclipse IDE with the 'JUnit' test runner. The 'JUnit' tab on the left indicates 'Finished after 0.131 seconds' with 'Runs: 1/1', 'Errors: 0', and 'Failures: 0'. The 'testing (Runner: JUnit 5) (0.019 s)' tab is active. The 'Failure Trace' is empty. The 'testing.java' file is open, showing the following code:

```
11  unittest obj1=new unittest();
12  String s1="abc";
13  String s2="abcd";
14  boolean output=obj1.prefix(s1,s2);
15  boolean expected=true;
16  assertEquals(expected,output);
17
18
19
20
21  @Test
22  public void test1() {
23      unittest obj2=new unittest();
24      String s1="abc";
25      String s2="ab";
26      boolean output=obj2.prefix(s1,s2);
27      boolean expected=true;
28      assertEquals(expected,output);
29
30  }
31
32  @Test
33  public void test2() {
34      unittest obj3=new unittest();
35      String s1="abc";
36      String s2="abef";
37      boolean output=obj3.prefix(s1,s2);
38      boolean expected=false;
39      assertEquals(expected,output);
40
41  }
42
43  @Test
44  public void test3() {
45      unittest obj4=new unittest();
46      String s1="abc";
47      String s2="abef";
48      boolean output=obj4.prefix(s1,s2);
49      boolean expected=false;
50      assertEquals(expected,output);
51
52  }
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
```

5. String s1 = "", s2 = "abef"; expected = true

Test case passed!



The screenshot shows the Eclipse IDE with the 'JUnit' test runner. The 'JUnit' tab on the left indicates 'Finished after 0.137 seconds' with 'Runs: 1/1', 'Errors: 0', and 'Failures: 0'. The 'testing (Runner: JUnit 5) (0.021 s)' tab is active. The 'Failure Trace' is empty. The 'testing.java' file is open, showing the following code:

```
22  unittest obj2=new unittest();
23  String s1="abc";
24  String s2="ab";
25  boolean output=obj2.prefix(s1,s2);
26  boolean expected=true;
27  assertEquals(expected,output);
28
29
30
31
32  @Test
33  public void test2() {
34      unittest obj3=new unittest();
35      String s1="abc";
36      String s2="abef";
37      boolean output=obj3.prefix(s1,s2);
38      boolean expected=false;
39      assertEquals(expected,output);
40
41  }
42
43  @Test
44  public void test3() {
45      unittest obj4=new unittest();
46      String s1="abc";
47      String s2="abef";
48      boolean output=obj4.prefix(s1,s2);
49      boolean expected=false;
50      assertEquals(expected,output);
51
52  }
53
54  @Test
55  public void test4() {
56      unittest obj5=new unittest();
57      String s1="";
58      String s2="abef";
59      boolean output=obj5.prefix(s1,s2);
60      boolean expected=true;
61      assertEquals(expected,output);
62
63  }
64
65
66
67
```

PROGRAM P6

Consider again the triangle classification program (P4) with a slightly different specification: The program reads floating values from the standard input. The three values A, B, and C are interpreted as representing the lengths of the sides of a triangle. The program then prints a message to the standard output that states whether the triangle, if it can be formed, is scalene, isosceles, equilateral, or right angled. Determine the following for the above program:

a) Identify the equivalence classes for the system

Equivalence Classes will contain:

1. All sides are positive, real numbers.
2. One or more sides are negative or zero.
3. The sum of the lengths of any two sides is less than or equal to the length of the remaining side.
4. The sum of the lengths of any two sides is greater than the length of the remaining side.

Examples

E1 : $a+b \leq c$ (point 3)

E2 : $a+c \leq b$ (point 3)

E3 : $b+c \leq a$ (point 3)

E4 : $a=b, b=c, c=a$

E5 : $a=b, a \neq c$

E6 : $a=c, a \neq b$

E7 : $b=c, b \neq a$

E8 : $a \neq b, b \neq c, c \neq a$

E9: $a^2 + b^2 = c^2$

E10: $b^2 + c^2 = a^2$

E11: $a^2 + c^2 = b^2$

E12 : $a+b > c$ (point 4)

E13: $a+c \geq b$ (point 4)

E14: $b+c \geq a$ (point 4)

b) Identify test cases to cover the identified equivalence classes. Also, explicitly mention which test case would cover which equivalence class.

Test cases

1. Right angled triangle (point 1 of (a)) - $A = 5, B = 12, C = 13$
2. Equilateral triangle (point 1 of (a)) - $A = 3, B = 3, C = 3$
3. Scalene triangle (point 1 of (a)) - $A = 3, B = 4, C = 3$
4. Isosceles triangle (point 1 of (a)) - $A = 3, B = 2, C = 3$
4. Invalid Input - $A = 1, B = 2, C = 3$
5. Invalid Input - $A = 0, B = 4, C = -1$

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

* Test cases

1. $A = 5, B = 4, C = 3$
2. $A = 5, B = 1, C = 6$
3. $A = 2, B = 3, C = 6$

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

* Test cases

1. $A = 4, B = 3, C = 4$
2. $A = 4, B = 3, C = 3.9$
3. $A = 4, B = 3, C = 4.1$

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

* Test cases

1. $A = 3, B = 3, C = 3$
2. $A = 3, B = 2.9, C = 3.1$

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

* Test cases

1. $A = 3, B = 4, C = 5$
2. $A = 6, B = 8, C = 10$

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

* Test cases

1. $A = 2, B = 2, C = 4$
2. $A = 2, B = 4, C = 2$

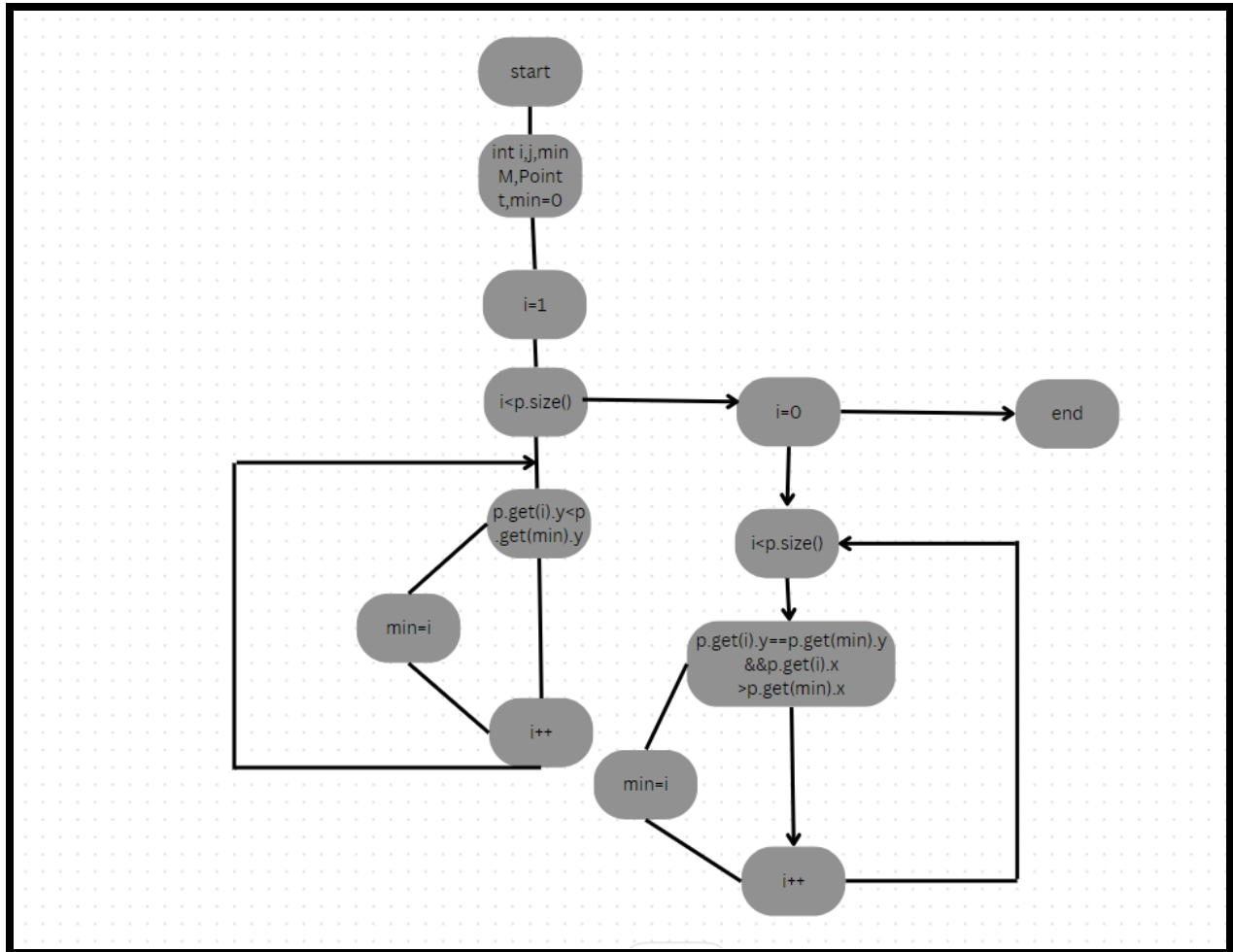
h) For non-positive input, identify test points.

* Test cases

1. $A = -3, B = 3, C = 4.5$
2. $A = 0, B = 4, C = 5$

SECTION B

1. Control Flow Graph



2. Construct test sets for your flow graph that are adequate for the following criteria:

- Statement Coverage.
- Branch Coverage.
- Basic Condition Coverage.

a) Statement coverage test sets:

* Test cases

- p is an empty vector
- p is a vector with one point
- p is a vector with two points having same y component

4. p is a vector with two points having different y components
5. p is a vector with three or more different points with different points with same y components
6. p is a vector with three or more different points with different points with different y components

b) Branch coverage test sets:

***TEST CASES**

1. p is an empty vector
 2. p is a vector with one point
 3. p is a vector with two points having same y component
 4. p is a vector with two points having different y components
 5. p is a vector with three or more different points with different points with same y components and with same x components.
 6. p is a vector with three or more different points with different points with different y components and with same x components.
 7. p is a vector with three or more points with the same x and y components
- </pre>

c) Basic condition coverage test sets:

***TEST CASES**

1. p is an empty vector
2. p is a vector with one point
3. p is a vector with two points having same y component
4. p is a vector with two points having different y components
5. p is a vector with three or more different points with different points with same y components and with same x components.
6. p is a vector with three or more different points with different points with different y components and with same x components.
7. p is a vector with three or more points with the same x and y components

8. p is a vector with some of them having same x component and all of them having same y component

*** Test cases examples:**

- 1) $p = [(x=2, y=3), (x=2, y=2), (x=1, y=5), (x=1, y=4)]$
- 2) $p = [(x=5, y=6), (x=3, y=2), (x=3, y=4), (x=1, y=2)]$
- 3) $p = [(x=5, y=6), (x=3, y=5), (x=1, y=5), (x=4, y=5), (x=2, y=7)]$
- 4) $p = [(x=7, y=8)]$
- 5) $p = []$

These 5 test cases covers all - statement coverage, branch coverage and basic condition coverage.