Software Engineering

LAB-7

Name: Jaimin Rathwa

Student ID:202001423

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 preset in an	Index v
V is not present	-1

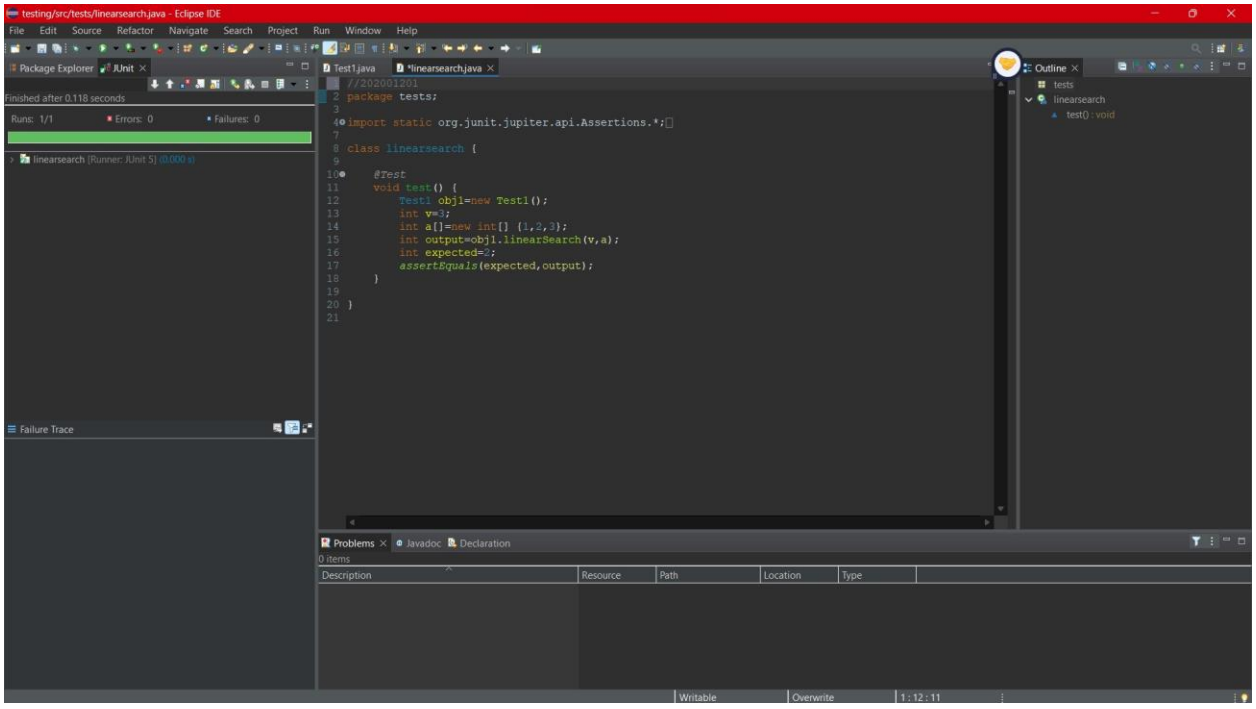
Boundary Value Analysis

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

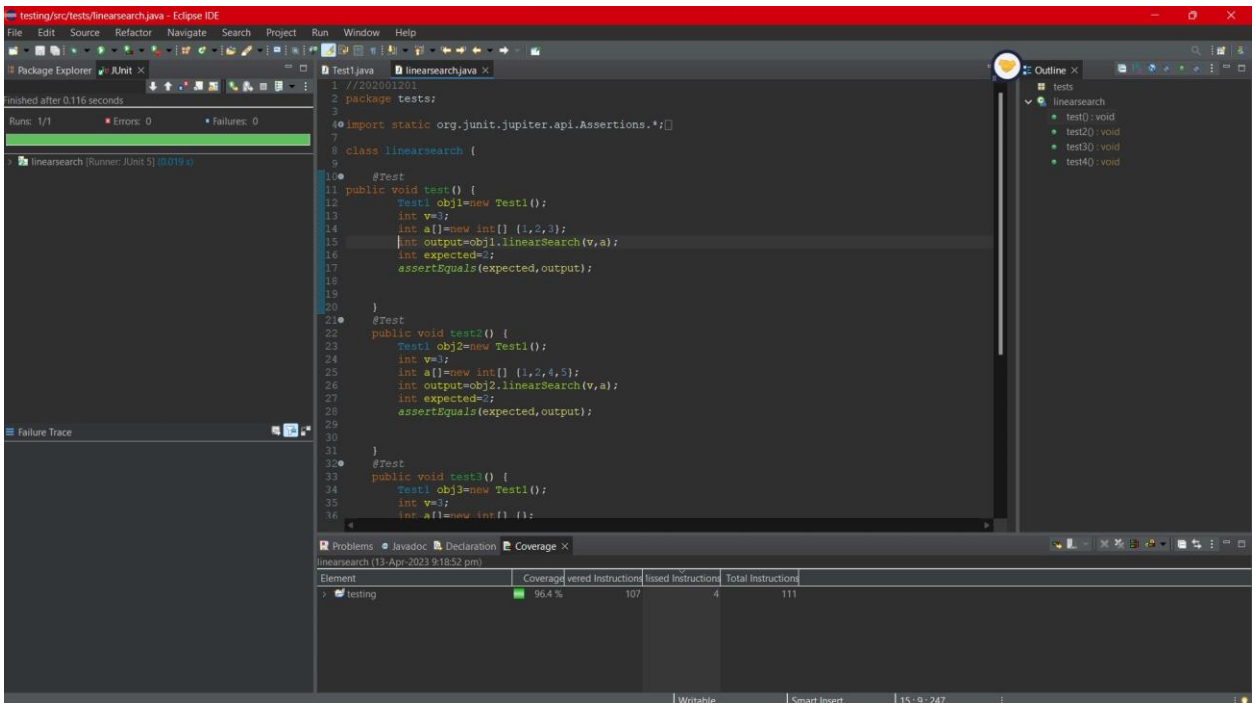
1. v = 3; a[] = {1,2,3}; expected = 2

Test case passed!



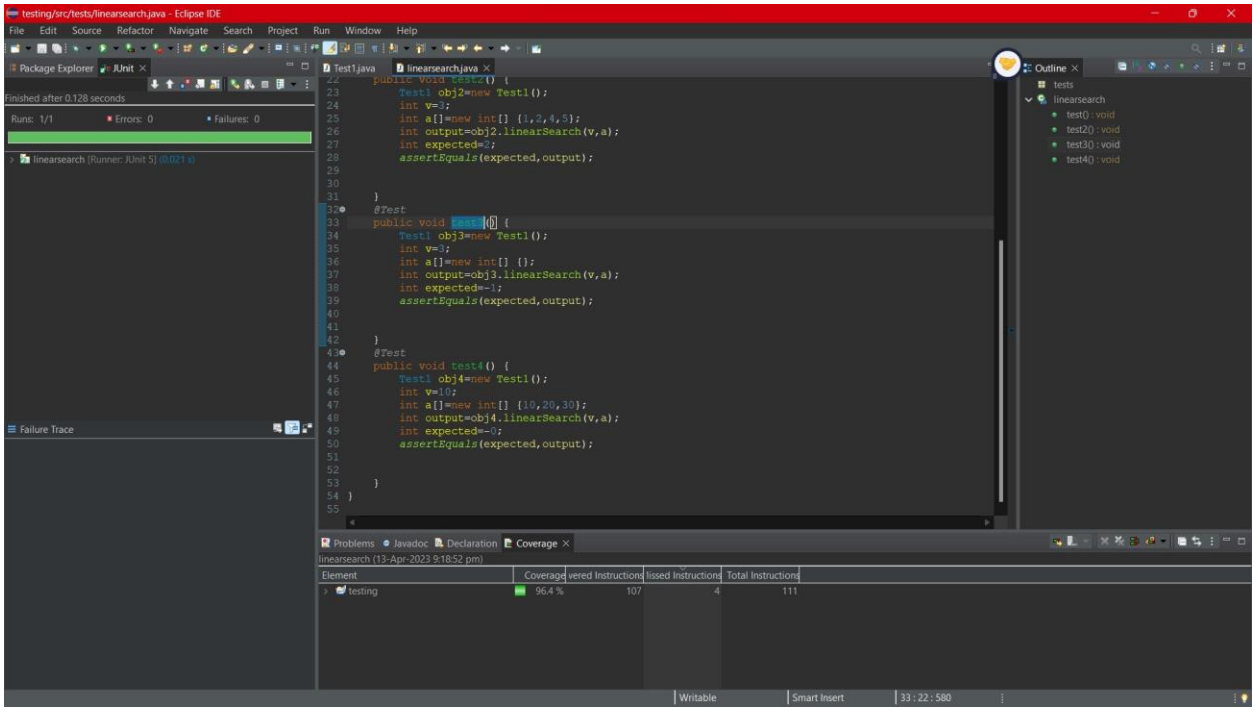
3. v = 3; a[] = {}; expected =-1;

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

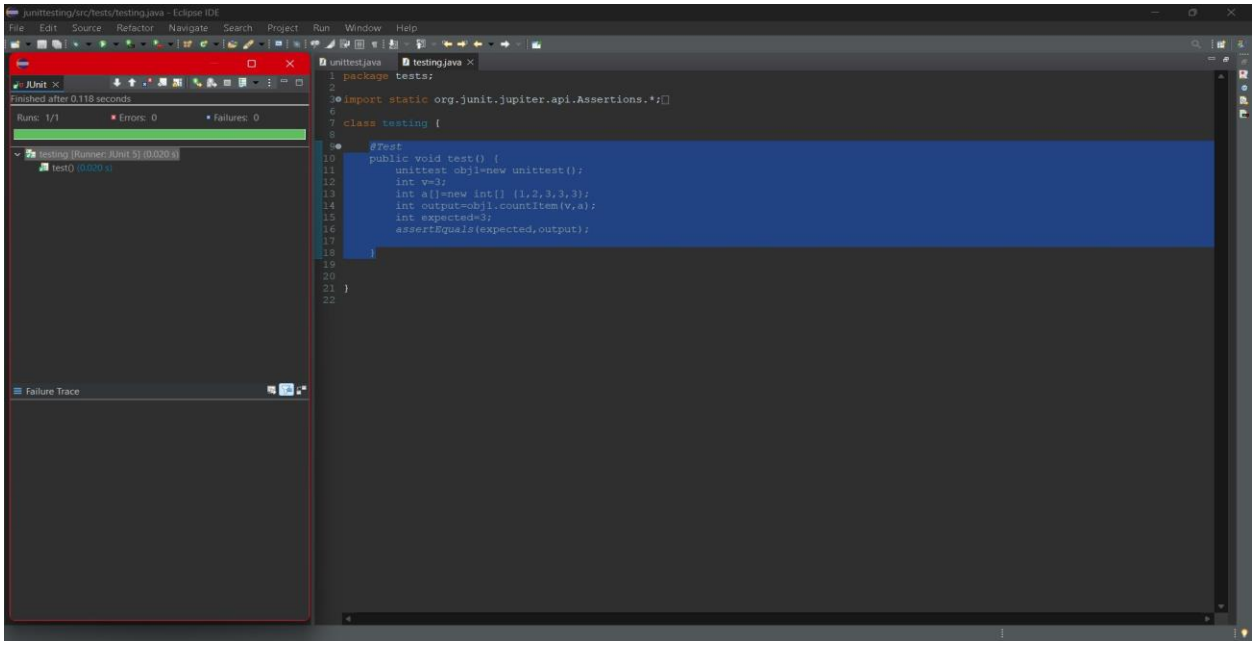
Empty array a	0
V is present once in a	1
V is present multiple times in a	Number of times V appears in a
V is present at the first index of a	1

V is present at the last index of a	1
V is not present in a	0

TEST CASES

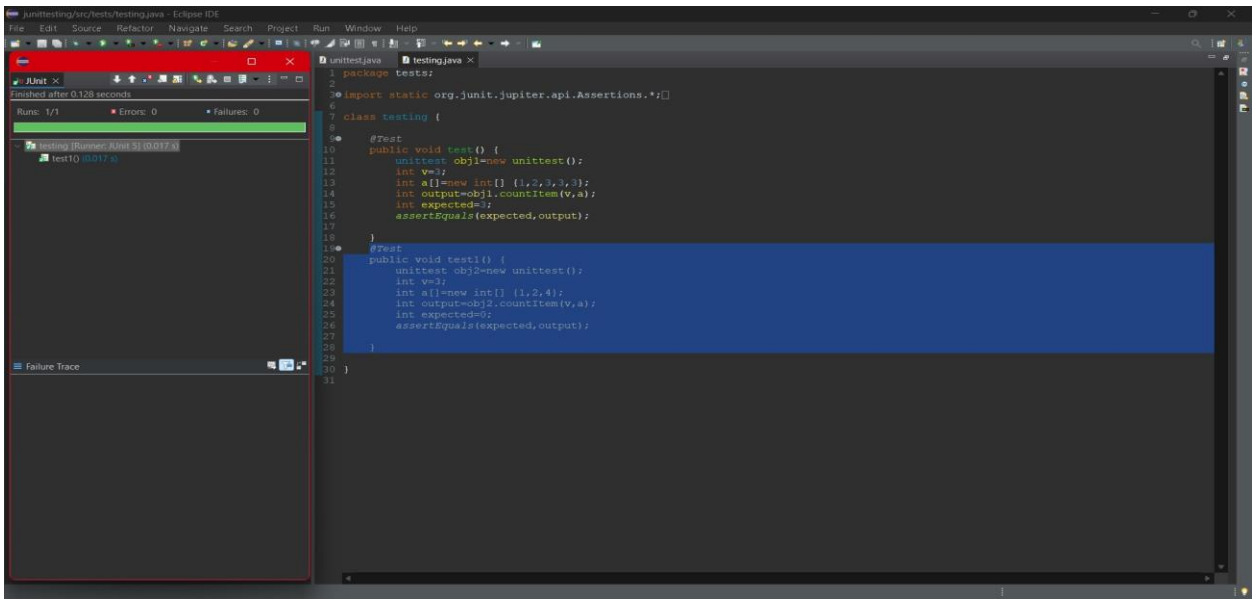
1. v = 3; a[] = {1,2,3,3,3}; expected = 3

   Test case passed!



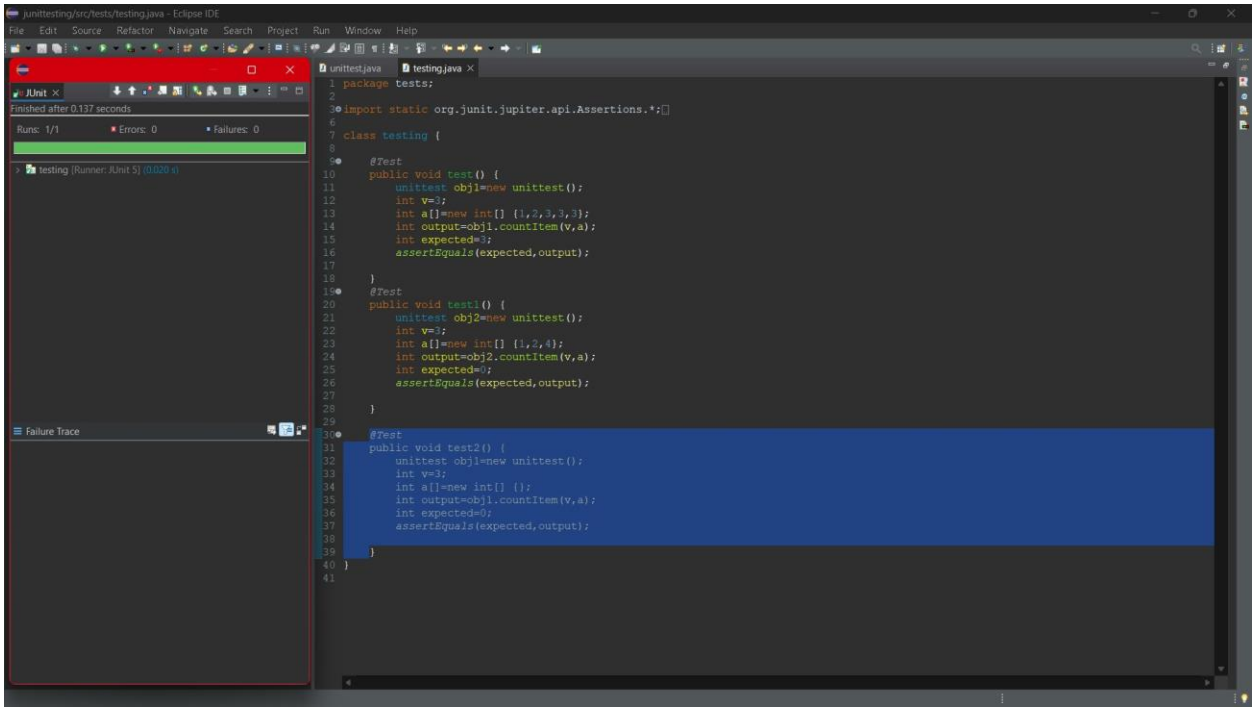
.

2. v = 3; a[] = {1,2,4}; expected = 0 Test case passed!



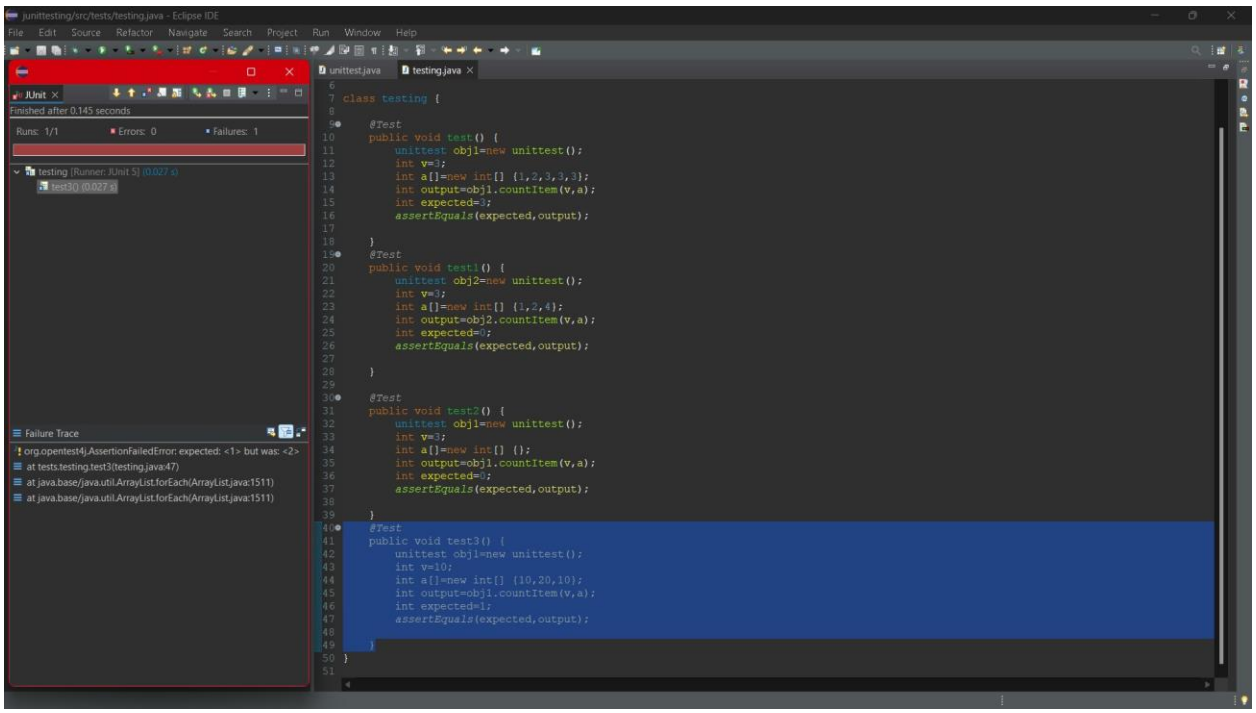
3. v = 3; a[] = {}; expected = 0 Test case

passed!



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

test case failed!



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

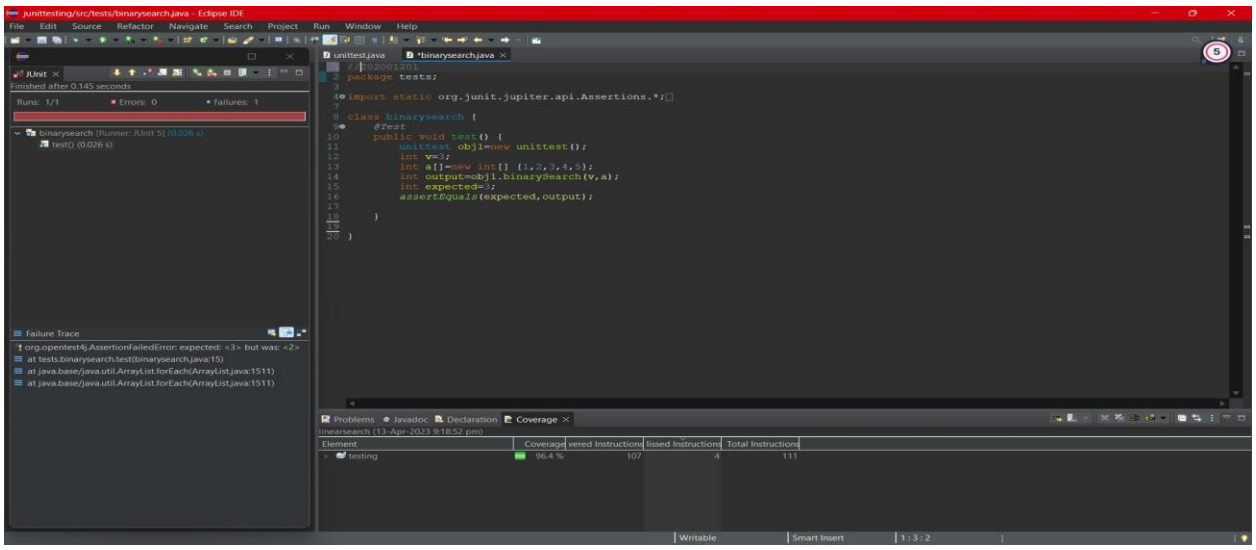
Boundary Value Analysis

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

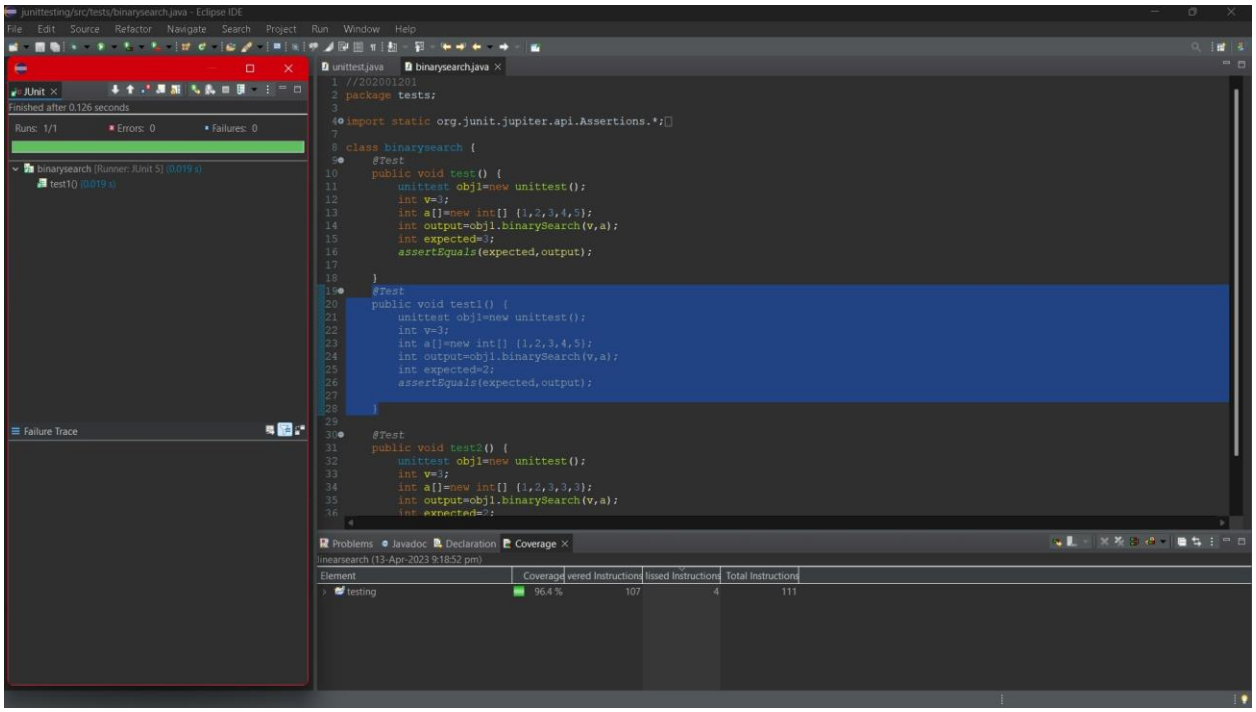
TEST CASES

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

   Test case failed!

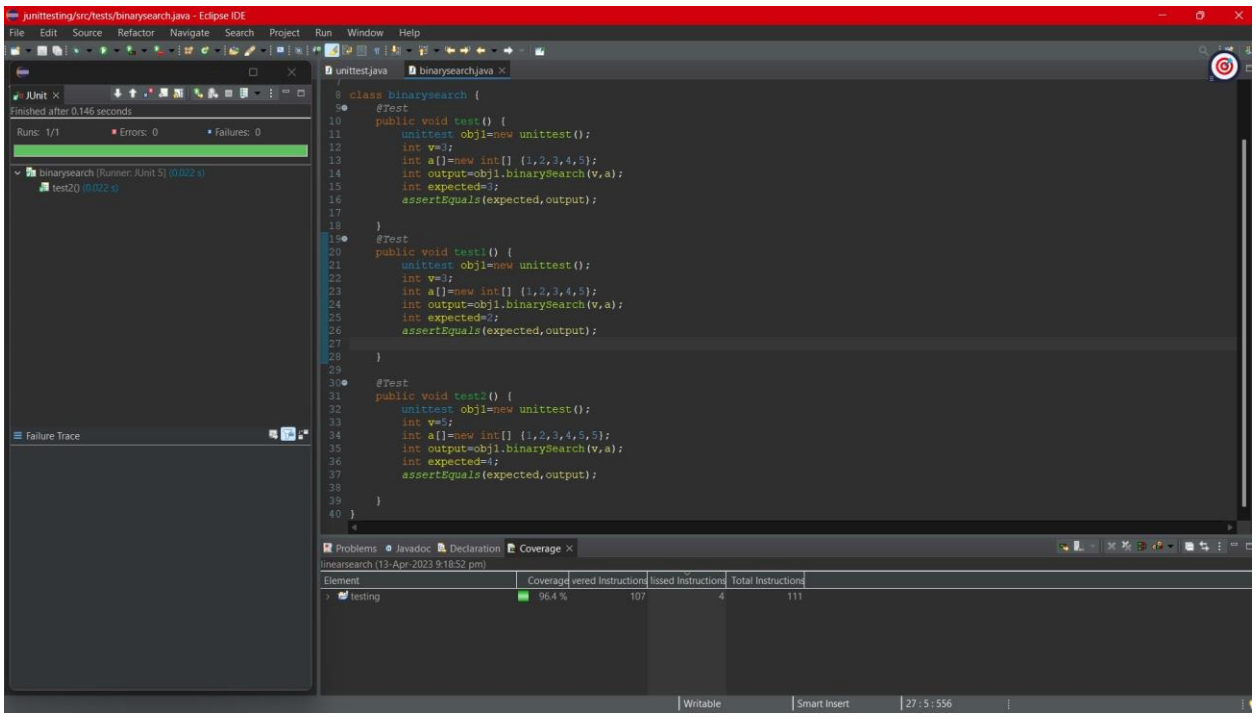


2. v = 3; a[] = {1,2,3,4,5}; expected = 2
   Test case passed!



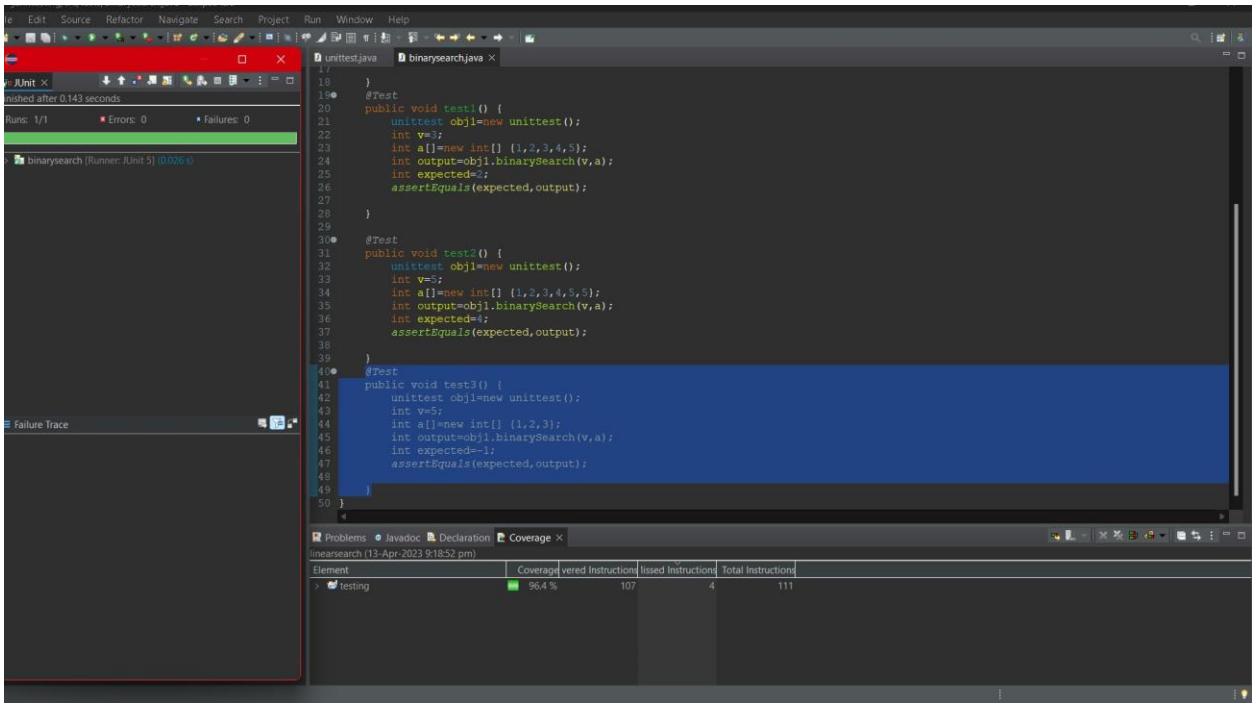
3. v = 5; a[] = {1,2,3,4,5,5}; expected = 4

Test case passed!



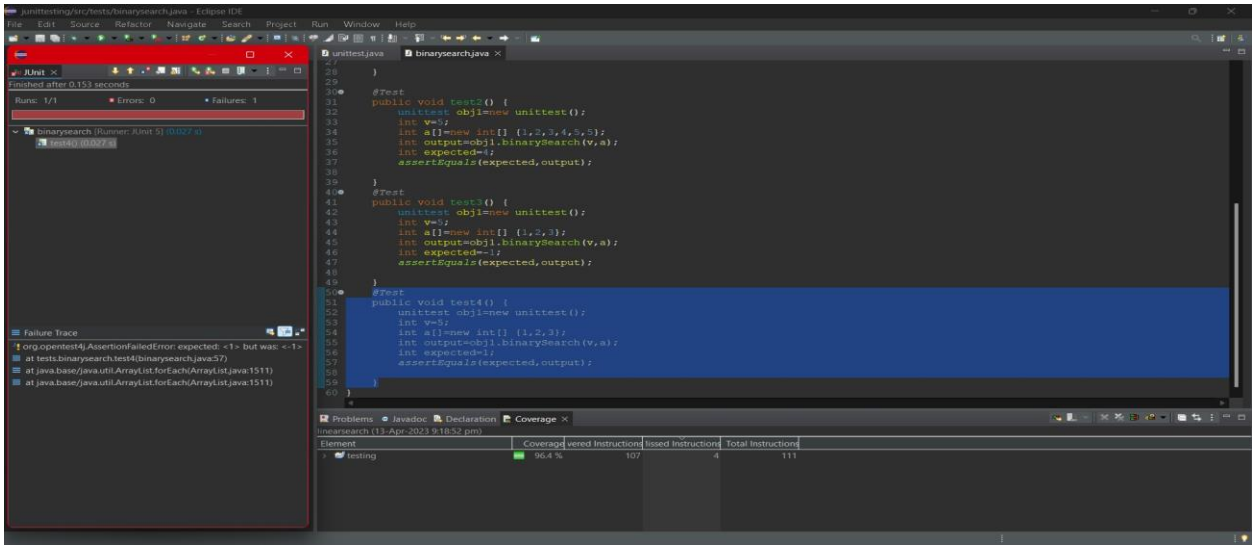
4. v = 5; a[] = {1,2,3}; expected = -1 Test

case passed!



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

Test case failed!



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<=c)	INVALID
Valid equilateral triangle(a=b=c)	EQAUILATERAL
Valid isosceles triangle(a=b<c)	ISOSCELES
Valid scalene triangle(a<b<c)	SCALENE

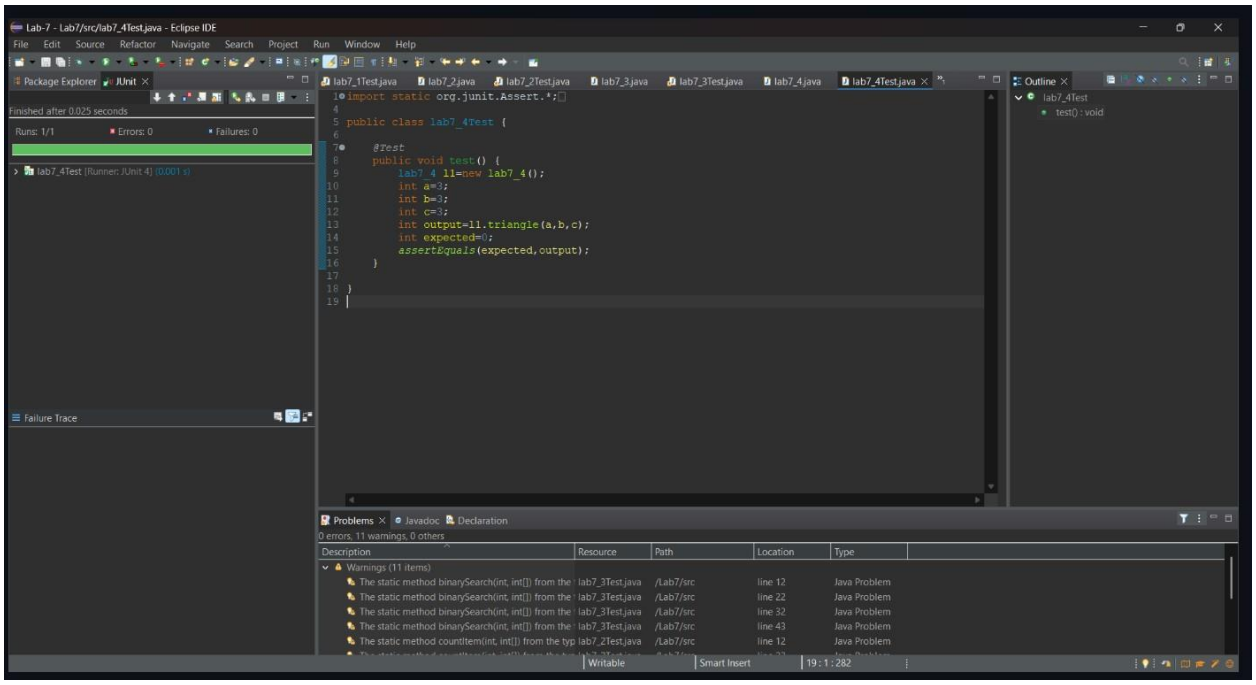
Boundary Value Analysis

Invalid triangle (a+b<=c)	INVALID
Invalid triangle(a+c<b)	INVALID
Invalid triangle(b+c<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 triable (a<b<c>)	SCALENE

TEST CASES

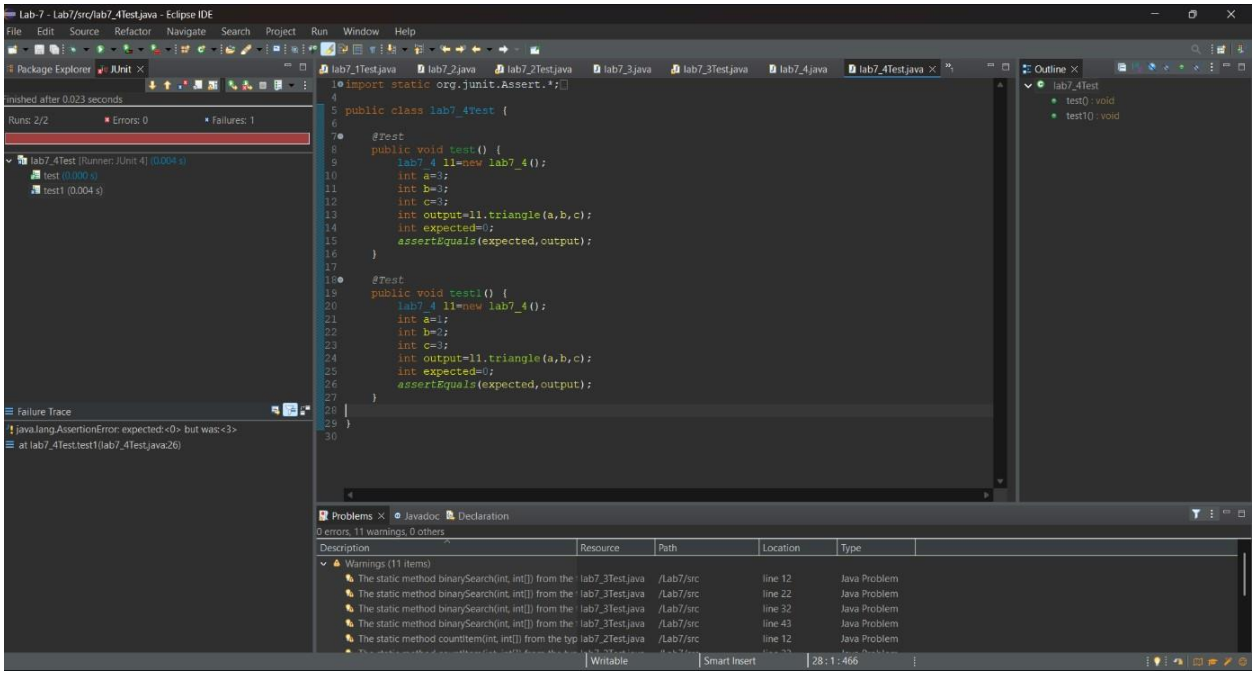
1. a = 3; b=3; c = 3; expected = 0
   (equilateral)

Test case passed!



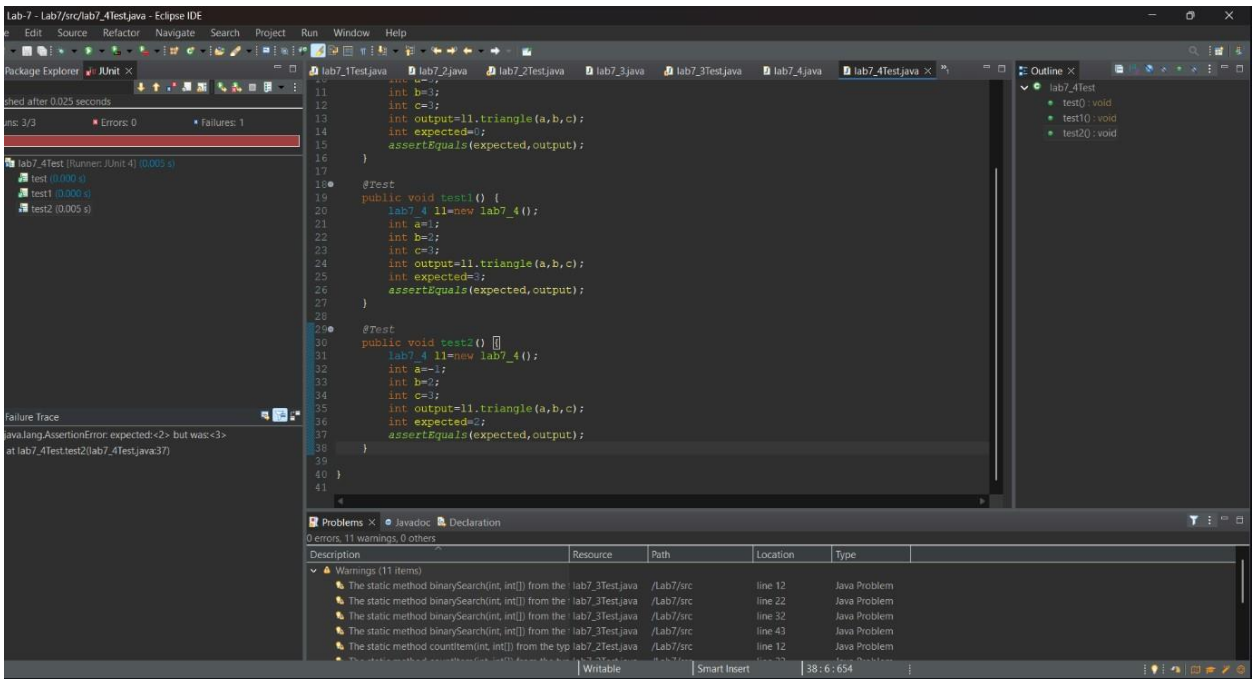
2. a = 1, b= 2, c= 3; expected = 0

Test case failed!



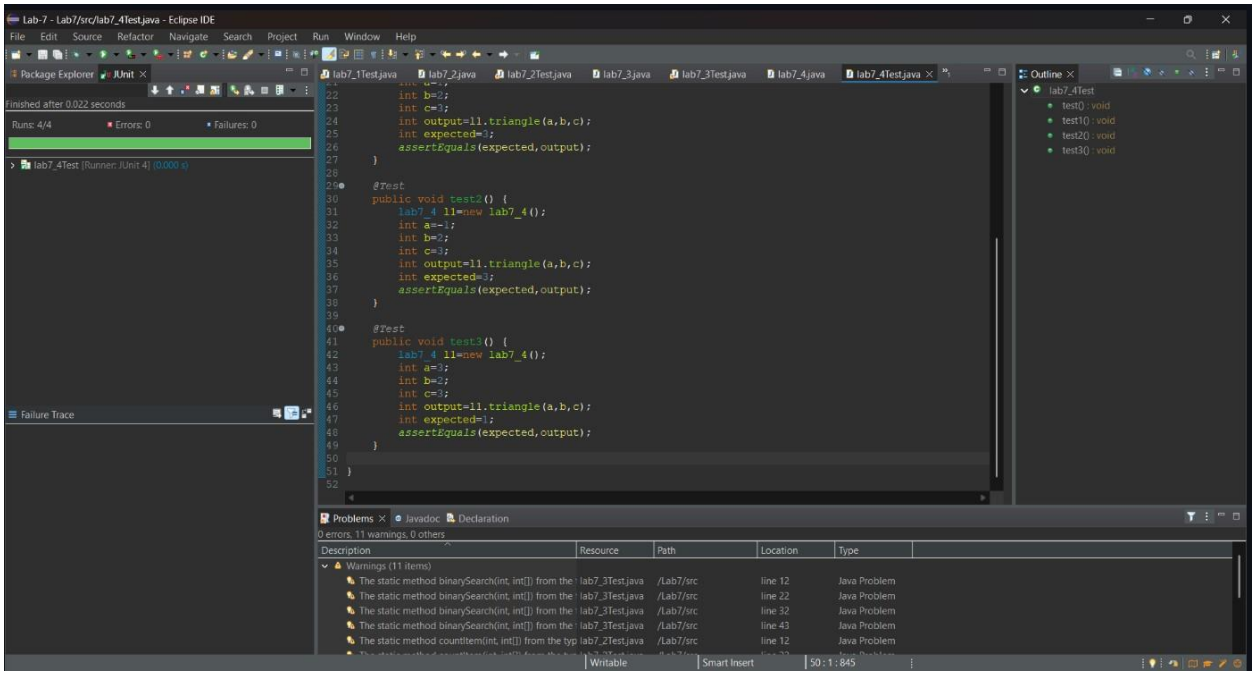
3. a = -1, b = 2, c = 3, expected = 2

Test case failed!



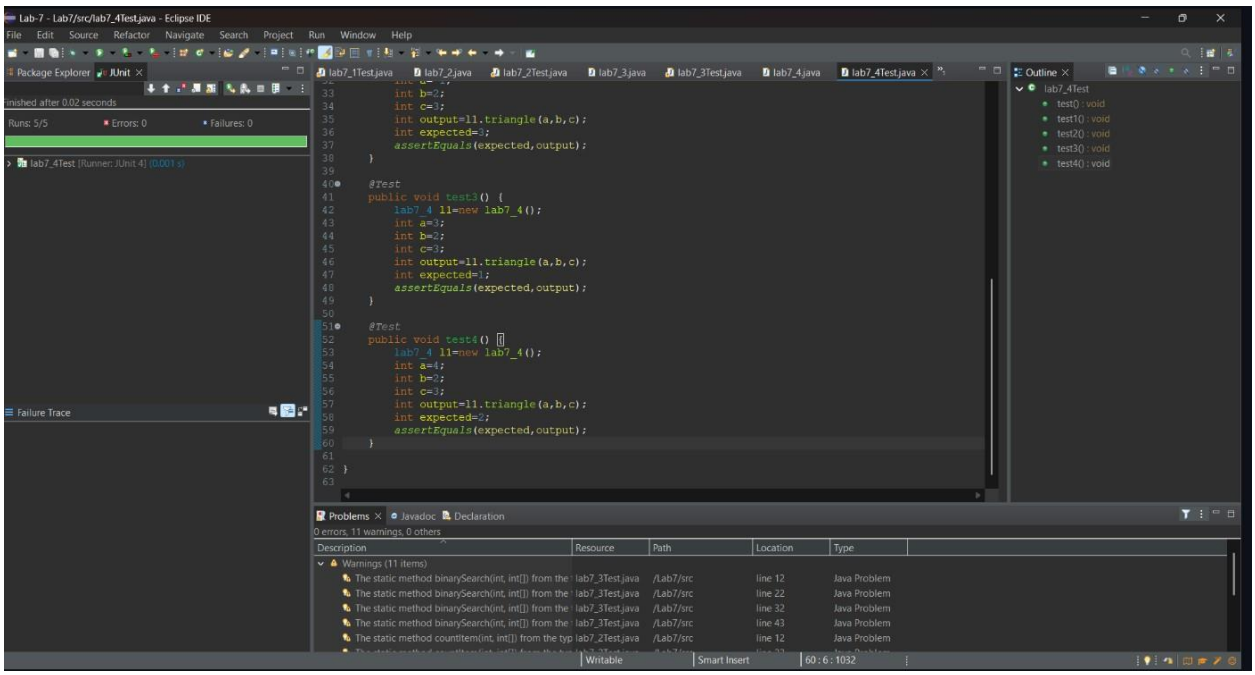
4. a = 3, b = 2, c = 3, expected = 1 Test

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)

  (s1.length() >
  s2.length())

return false;

for (int i = 0; i < s1.length(); i++)

  (s1.charAt(i) != s2.charAt(i))

return false;

return true;
```

```
if
{

}

{
if
{
```

}
}

}

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

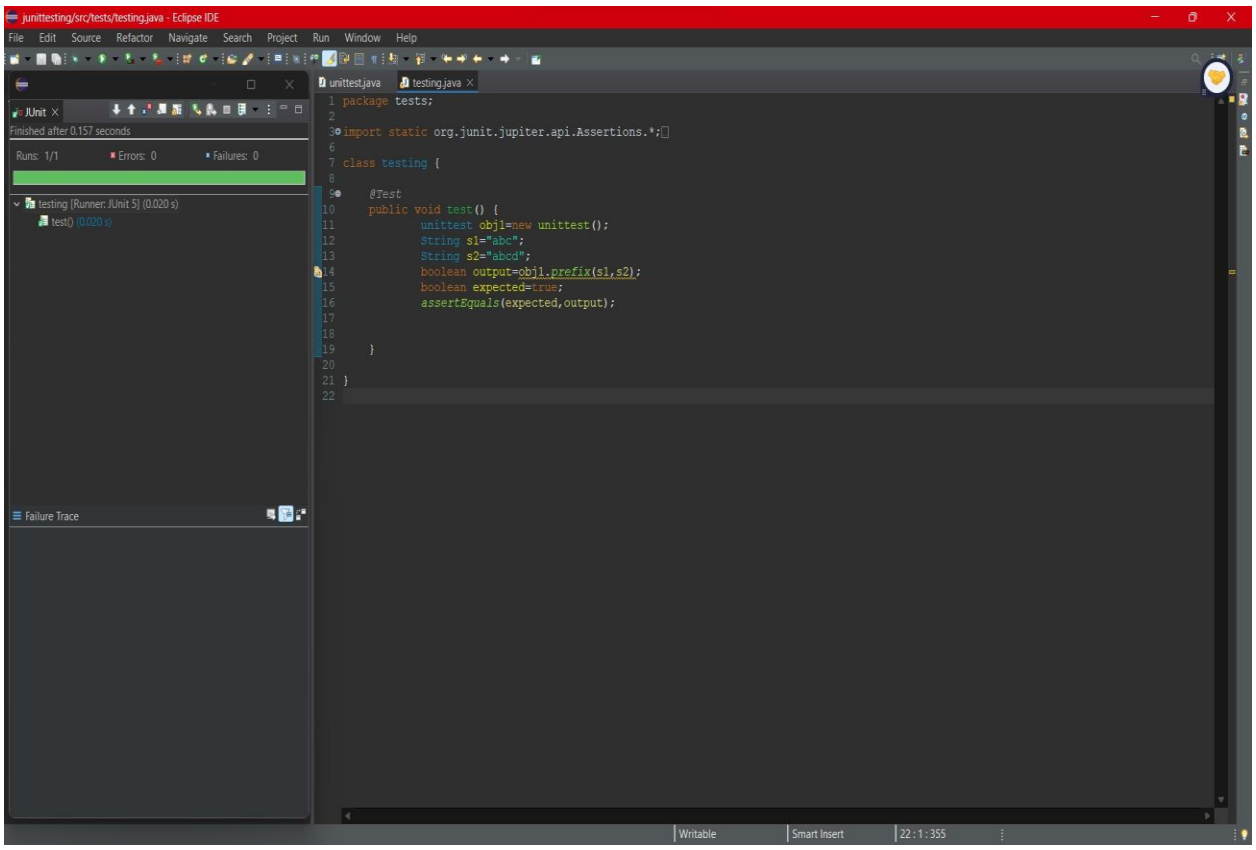
Boundary value analysis

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

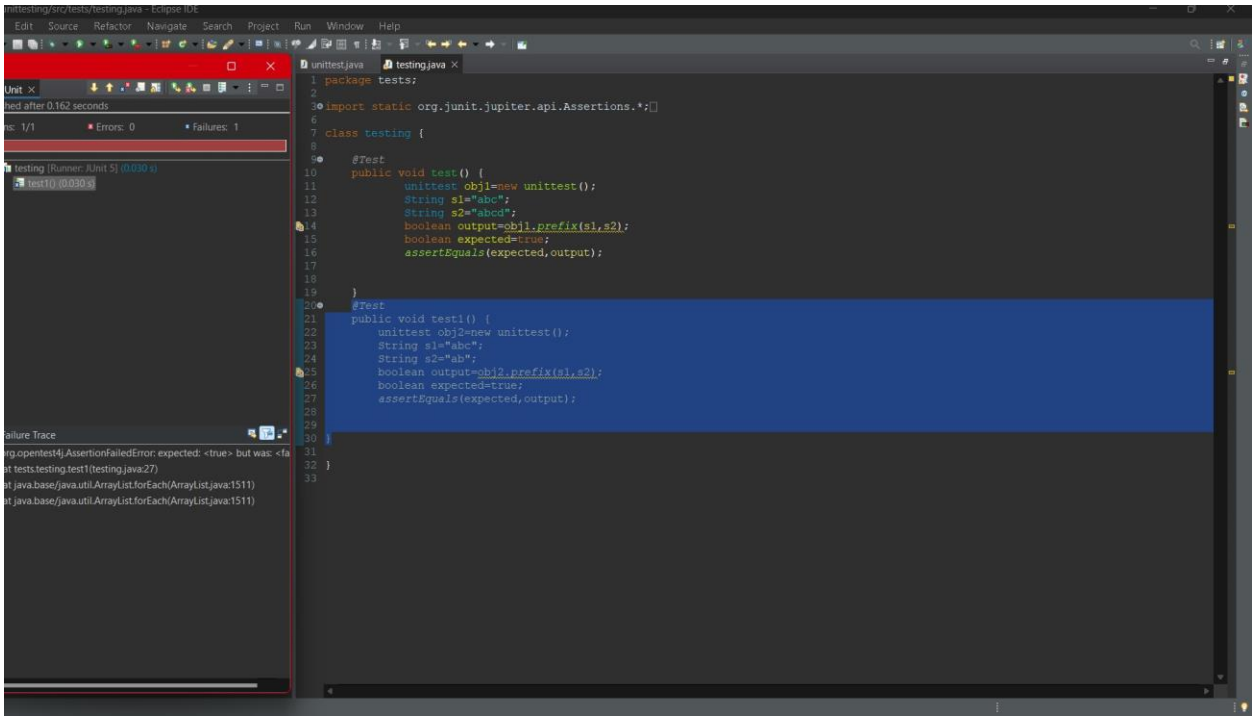
TEST CASES

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

Test case passed!

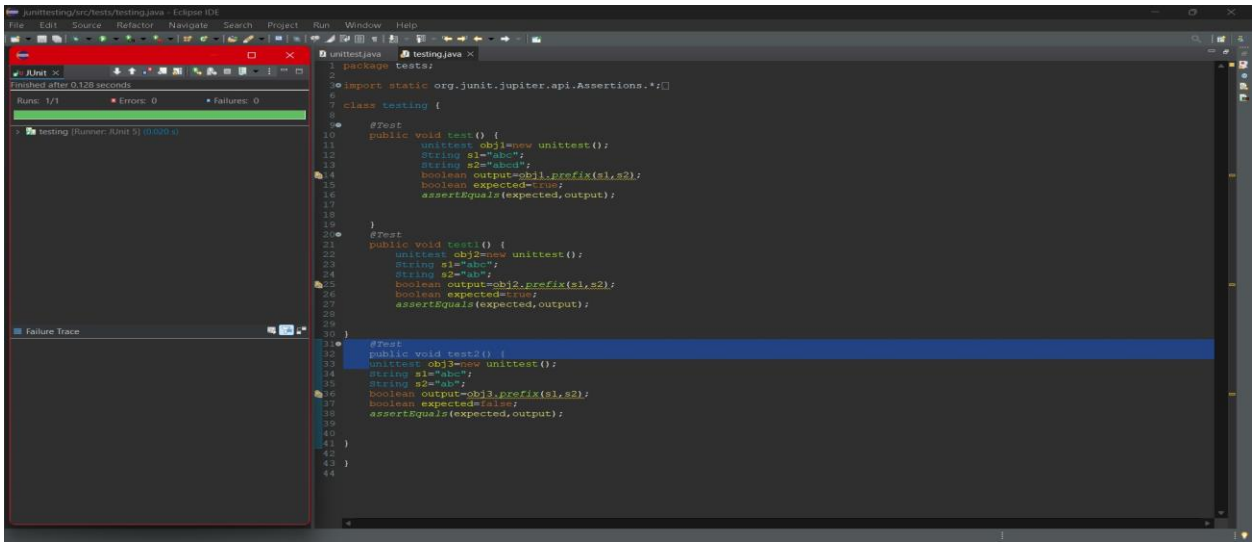


2. String s1 = "abc", s2 = "ab"; expected = true Test case failed!

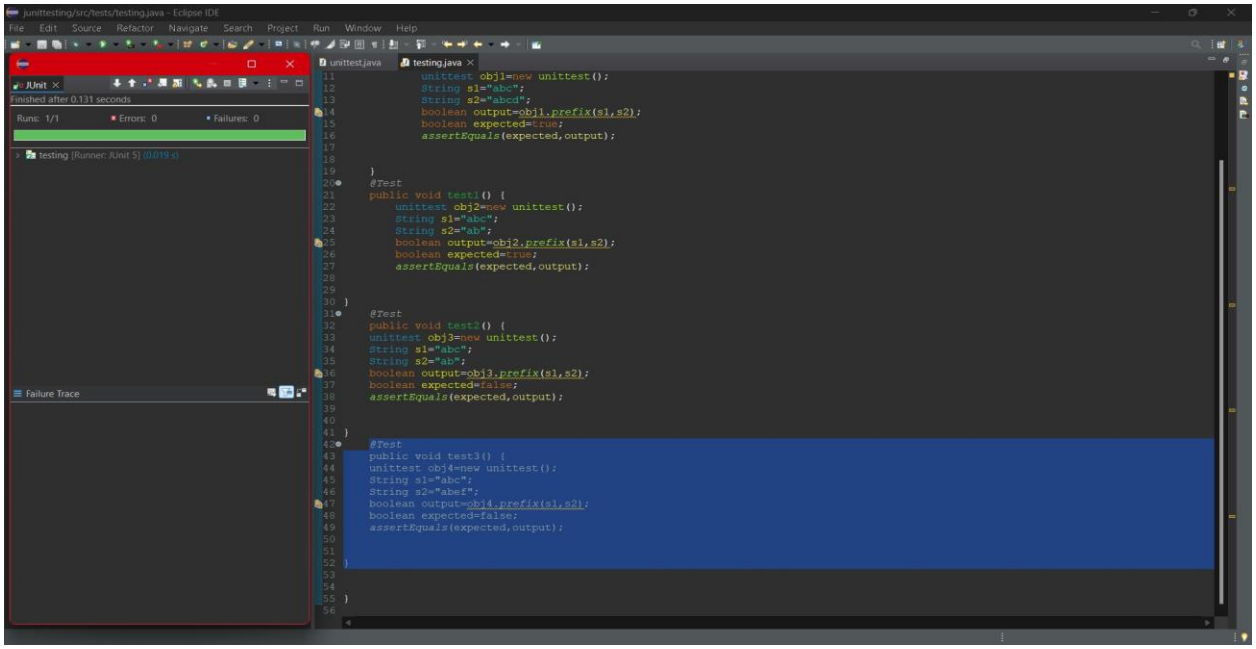


3. String s1 = "abc", s2 = "ab"; expected = false

Test case passed!

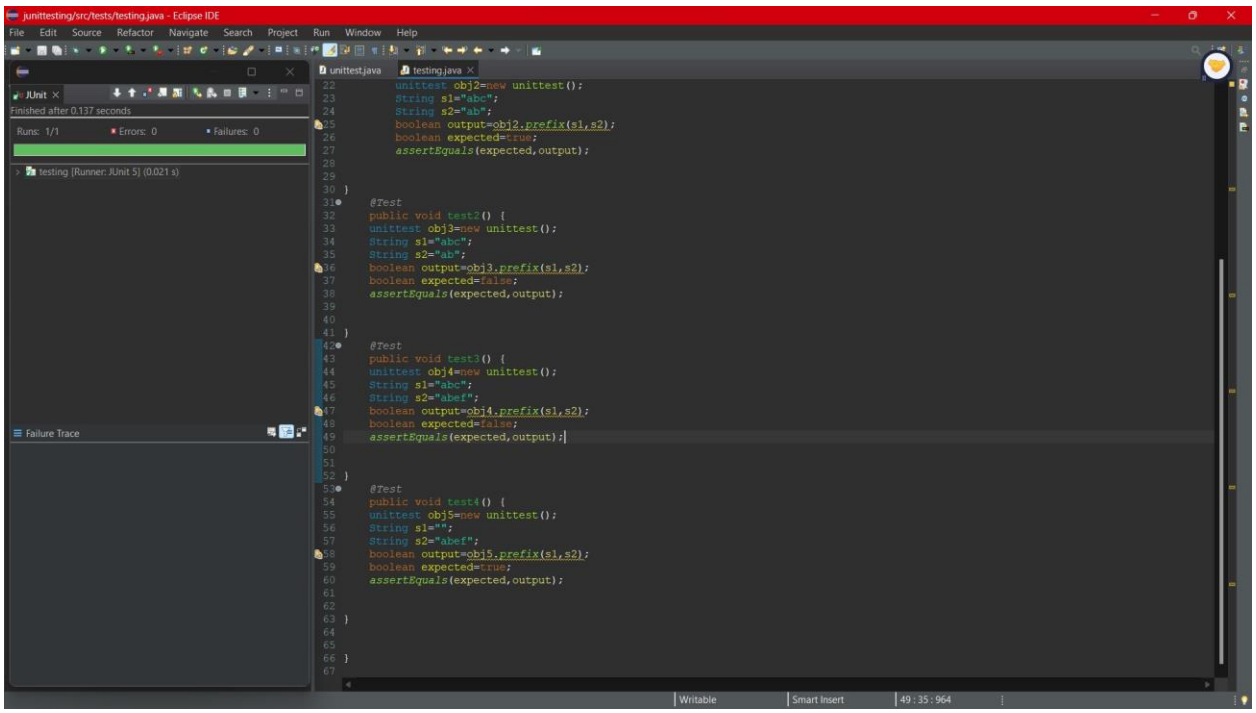


4. String s1 = "abc", s2 = "abef"; expected = false Test case passed!



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

Test case passed!



PROGRAM P6

Consider again the triangle classification program (P4) with a slightly different specification: The programreads 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<=c (point 3)
E2 : a+c<=b (point 3)
E3 : b+c<=a (point 3)
E4 : a=b, b=c, c=a
E5 : a=b, a!=c
E6 : a=c, a!=b
E7 : b=c, b!=a
E8 : a!=b ,b!=c, c!=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>=b (point 4)
E14: b+c>=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 A2 + B2 = C2 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

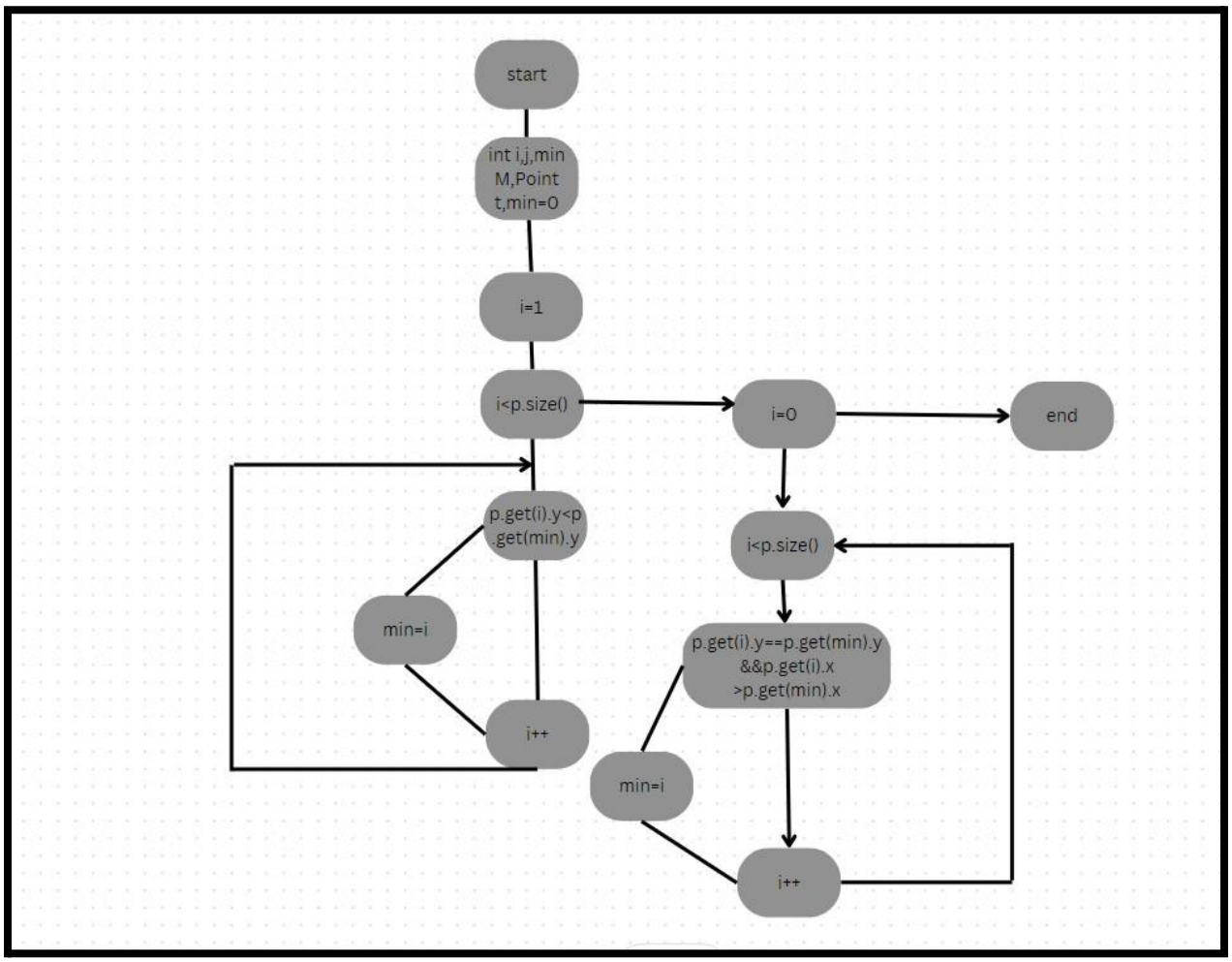
h) For non-positive input, identify test points.

* 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:**

a. Statement Coverage.
b. Branch Coverage.
c. Basic Condition Coverage.

a) Statement 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
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.