IT314 - Lab 7

Name: Rudra Gohel

ID: 202001178

Lab Date: 13 / 4 / 2023

Section A:

Inputs and their range:

1.) Day: [1,31]

2.) Month: [1, 12]

3.) Year: [1900, 2015]

Equivalence Class Partitioning:

ID	Input Date	Expected Output
E1	(15,10, 2022)	(14,10, 2022)
E2	(1, 6, 2018)	(31,5,2018)
E3	(31, 3, 2000)	30, 3, 2000
E4	(29, 2, 2022)	Invalid Date
E5	(31, 4, 2011)	Invalid Date
E6	(30, 2, 2000)	Invalid Date
E7	(0, 5, 2010)	Invalid Date
E8	(15, 13, 2005)	Invalid Date
E9	(31, 12, 1899)	Invalid Date

Boundary Value Analysis:

Input Date	Reason	Expected Output
(1, 1, 1900)	The earliest possible date	Invalid date
(31, 12, 2015)	The latest possible date	30, 12, 2015
(1, 2, 2000)	The earliest day of each month	31, 12, 1999
(31, 3, 2000)	The latest day of each month	30, 1, 2000
(29, 2, 2000)	Leap year day	28, 2, 2000
(29, 2, 1900)	Invalid leap year day	Invalid date
(31, 12, 1899)	One day before the earliest date	Invalid date
(1, 1, 2016)	One day after the latest date	Invalid date

P1. linearSearch

Java Code:

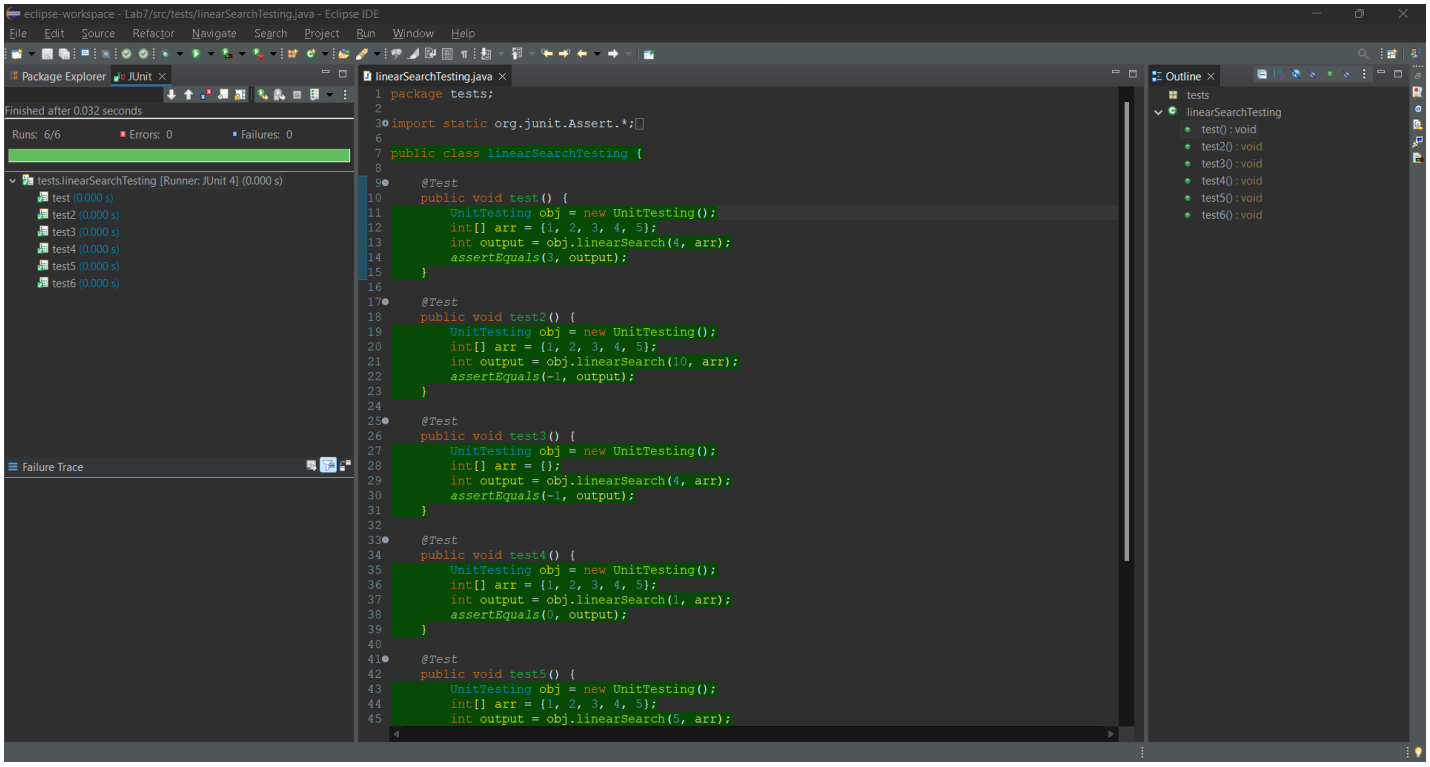
```
int linearSearch(int v, int a[])
{
      int i = 0;
      while (i < a.length)
      {
            if (a[i] == v)
                  return(i);
            i++;
      }
      return (-1);
}
```

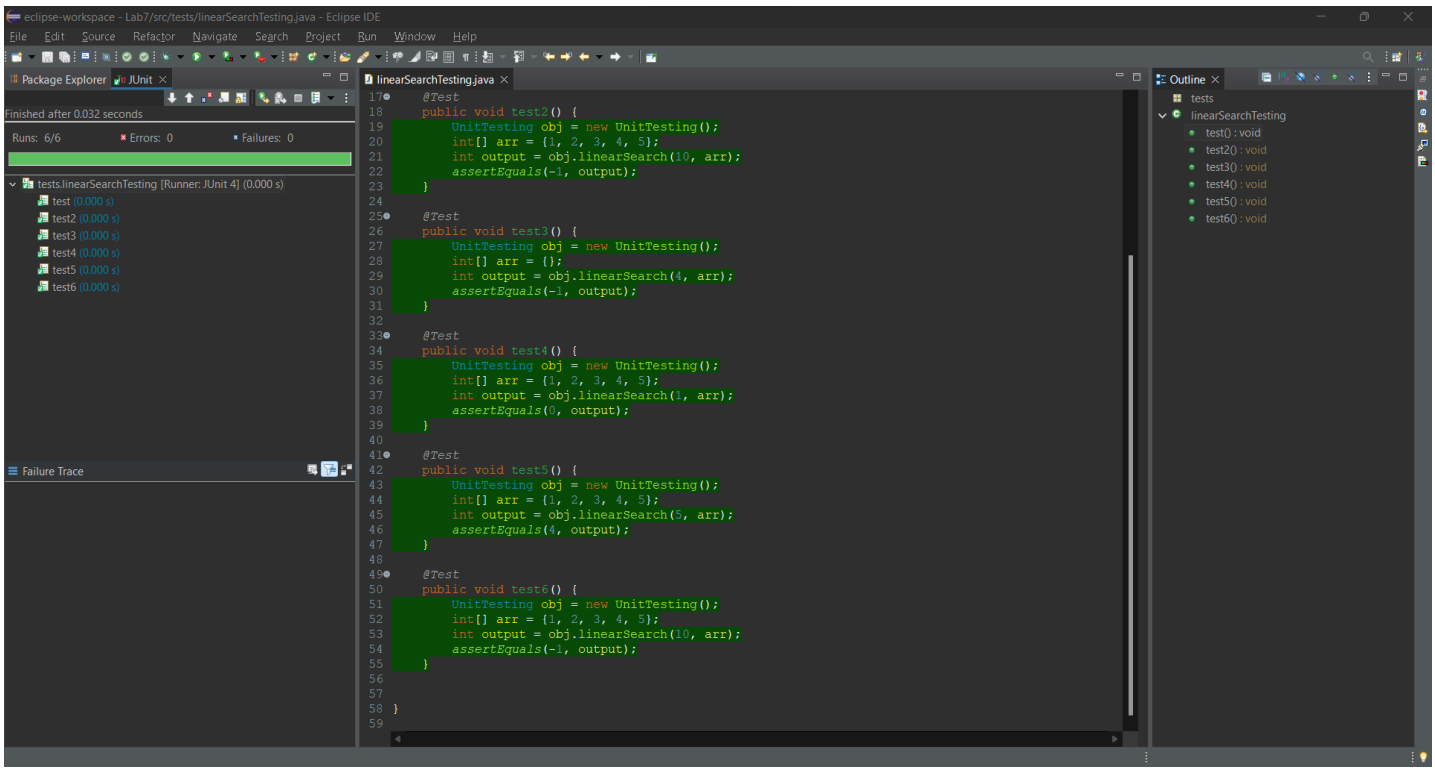
Equivalence Classes:

Input	Expected Output	a	v	Actual Output
v is present in a	v is not present in a	[1, 2, 3, 4, 5]	4	3
v is not present in a	-1	[1, 2, 3, 4, 5]	10	-1

Boundary Value Analysis:

Input	Expected Output	a	v	Actual Output
Empty array a	-1	[]	4	-1
v is present at the first index of a	0	[1, 2, 3, 4, 5]	1	-1
v is present at the last index of the length of a	last index	[1, 2, 3, 4, 5]	5	4
v is not present in a	-1	[1, 2, 3, 4, 5]	10	-1


eclipse-workspace - Lab7/src/tests/linearSearchTesting.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit
Finished after 0.032 seconds
Runs: 6/6 Errors: 0 Failures: 0
tests.linearSearchTesting [Runner: JUnit 4] (0.000 s)
test (0.000 s)
test2 (0.000 s)
test3 (0.000 s)
test4 (0.000 s)
test5 (0.000 s)
test6 (0.000 s)
Failure Trace
linearSearchTesting.java
1 package tests;
2
3 import static org.junit.Assert.*;
4
5 public class linearSearchTesting {
6
7 @Test
8 public void test() {
9 UnitTesting obj = new UnitTesting();
10 int[] arr = {1, 2, 3, 4, 5};
11 int output = obj.linearSearch(4, arr);
12 assertEquals(3, output);
13 }
14
15 @Test
16 public void test2() {
17 UnitTesting obj = new UnitTesting();
18 int[] arr = {1, 2, 3, 4, 5};
19 int output = obj.linearSearch(10, arr);
20 assertEquals(-1, output);
21 }
22
23 @Test
24 public void test3() {
25 UnitTesting obj = new UnitTesting();
26 int[] arr = {};
27 int output = obj.linearSearch(4, arr);
28 assertEquals(-1, output);
29 }
30
31 @Test
32 public void test4() {
33 UnitTesting obj = new UnitTesting();
34 int[] arr = {1, 2, 3, 4, 5};
35 int output = obj.linearSearch(1, arr);
36 assertEquals(0, output);
37 }
38
39 @Test
40 public void test5() {
41 UnitTesting obj = new UnitTesting();
42 int[] arr = {1, 2, 3, 4, 5};
43 int output = obj.linearSearch(5, arr);
44 assertEquals(4, output);
45 }
Outline
tests
linearSearchTesting
test() : void
test2() : void
test3() : void
test4() : void
test5() : void
test6() : void


eclipse-workspace - Lab7/src/tests/linearSearchTesting.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit
Finished after 0.032 seconds
Runs: 6/6 Errors: 0 Failures: 0
tests.linearSearchTesting [Runner: JUnit 4] (0.000 s)
test (0.000 s)
test2 (0.000 s)
test3 (0.000 s)
test4 (0.000 s)
test5 (0.000 s)
test6 (0.000 s)
Failure Trace
linearSearchTesting.java
17 @Test
18 public void test2() {
19 UnitTesting obj = new UnitTesting();
20 int[] arr = {1, 2, 3, 4, 5};
21 int output = obj.linearSearch(10, arr);
22 assertEquals(-1, output);
23 }
24
25 @Test
26 public void test3() {
27 UnitTesting obj = new UnitTesting();
28 int[] arr = {};
29 int output = obj.linearSearch(4, arr);
30 assertEquals(-1, output);
31 }
32
33 @Test
34 public void test4() {
35 UnitTesting obj = new UnitTesting();
36 int[] arr = {1, 2, 3, 4, 5};
37 int output = obj.linearSearch(1, arr);
38 assertEquals(0, output);
39 }
40
41 @Test
42 public void test5() {
43 UnitTesting obj = new UnitTesting();
44 int[] arr = {1, 2, 3, 4, 5};
45 int output = obj.linearSearch(5, arr);
46 assertEquals(4, output);
47 }
48
49 @Test
50 public void test6() {
51 UnitTesting obj = new UnitTesting();
52 int[] arr = {1, 2, 3, 4, 5};
53 int output = obj.linearSearch(10, arr);
54 assertEquals(-1, output);
55 }
56
57
58 }
59
Outline
tests
linearSearchTesting
test() : void
test2() : void
test3() : void
test4() : void
test5() : void
test6() : void

P2. countItem

```
int countItem(int v, int a[])

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

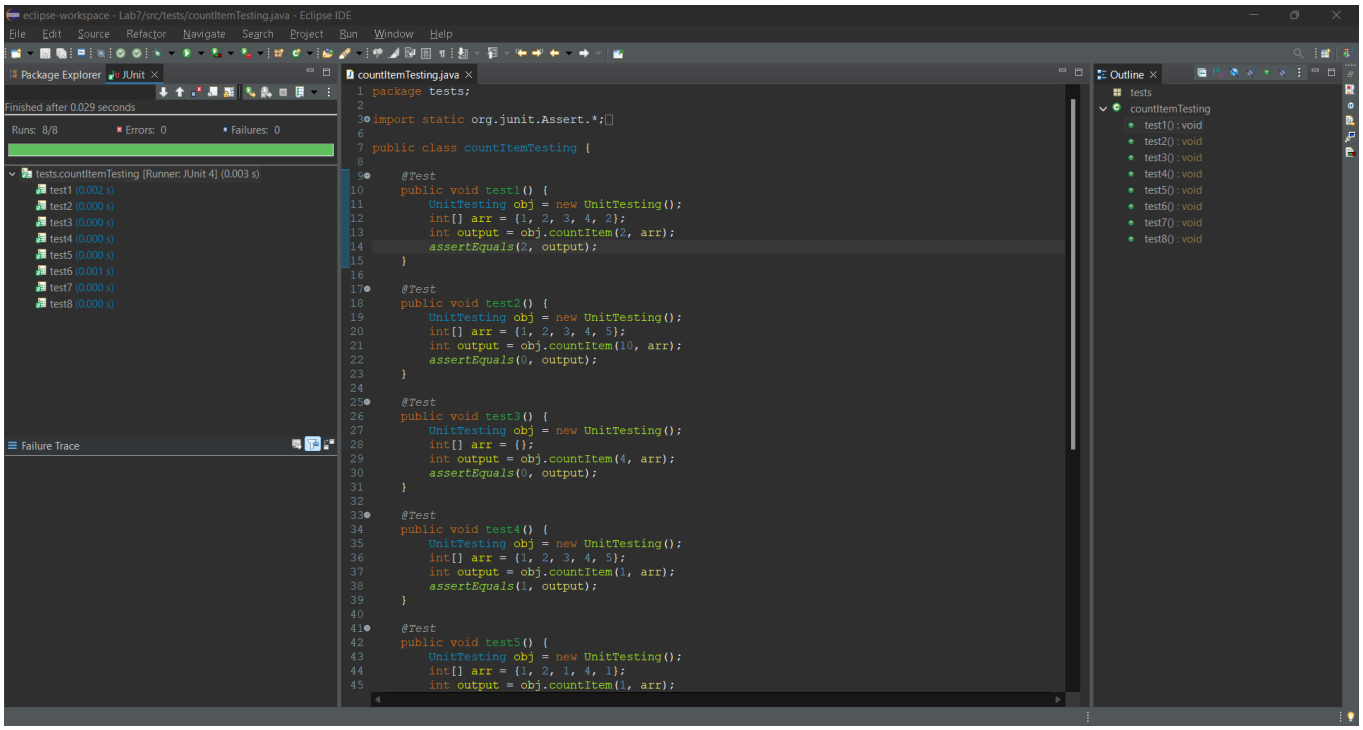
Equivalence Classes:

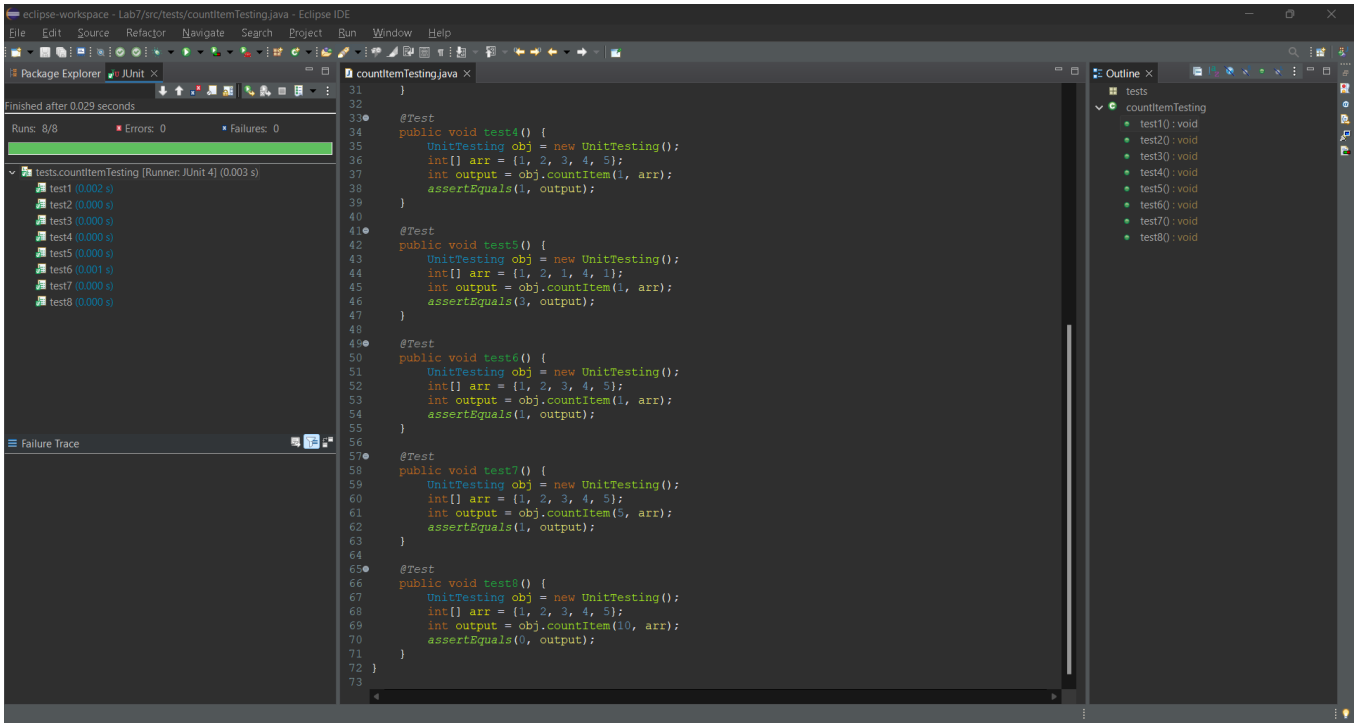
Input	Expected Output	a	v	Actual Output
v is present in a	Number of times v appears in a	[1, 2, 3, 4, 2]	2	2
v is not present in a	0	[1, 2, 3, 4, 5]	10	0

Boundary Value Analysis:

Input	Expected Output	a	v	Actual Output
Empty array a	0	[]	4	0
v is present once in a	1	[1, 2, 3, 4, 5]	1	1
v is present multiple times in a	Number of times v appears in a	[1, 2, 1, 4,1]	1	3
v is present at the first index of a	1	[1, 2, 3, 4, 5]	1	1

v is present at the last index of a	1	[1, 2, 3, 4, 5]	5	1
v is not present in a	0	[1, 2, 3, 4, 5]	10	0





P3. binarySearch

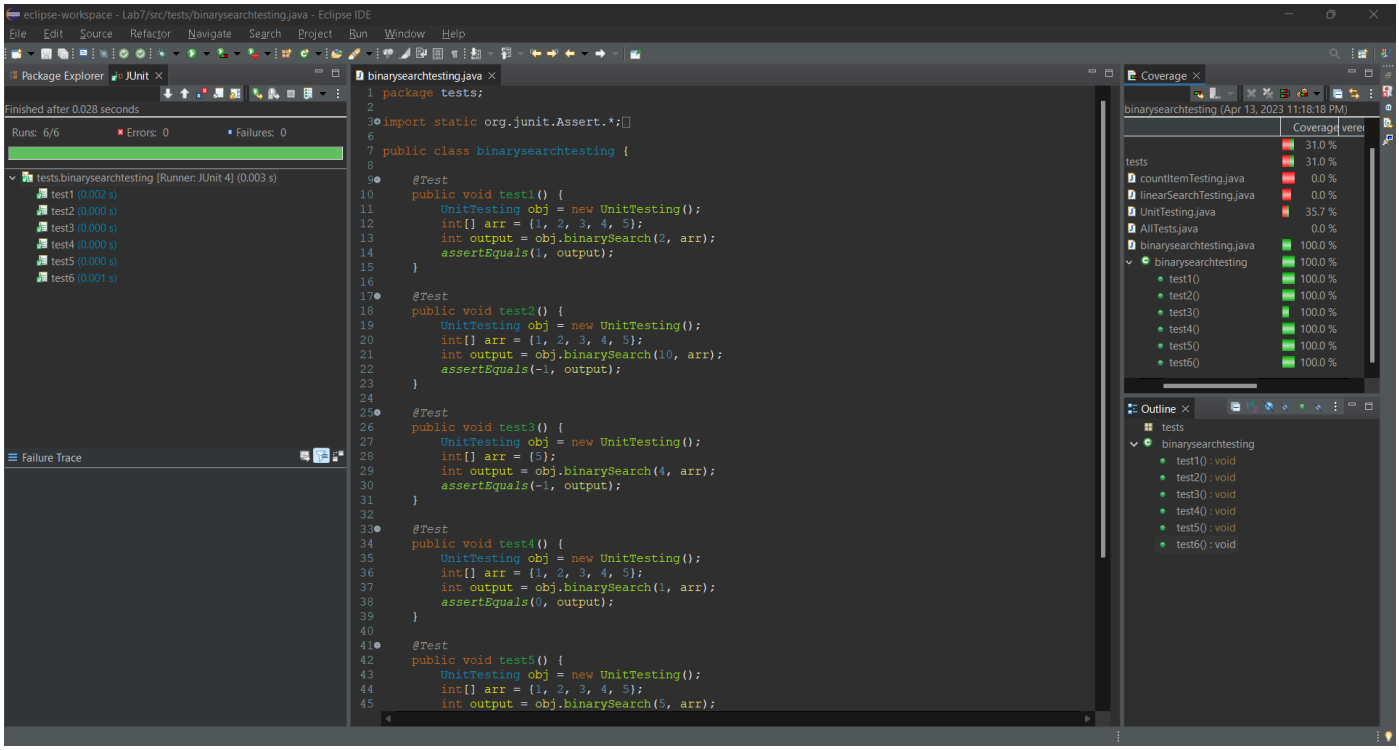
```
int binarySearch(int v, int a[])
{
      int lo,mid,hi;
      lo = 0;
      hi = a.length-1;
      while (lo <= hi)
      {
            mid = (lo+hi)/2;
            if (v == a[mid])
                  return (mid);
            else if (v < a[mid])
                  hi = mid-1;
            else
  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

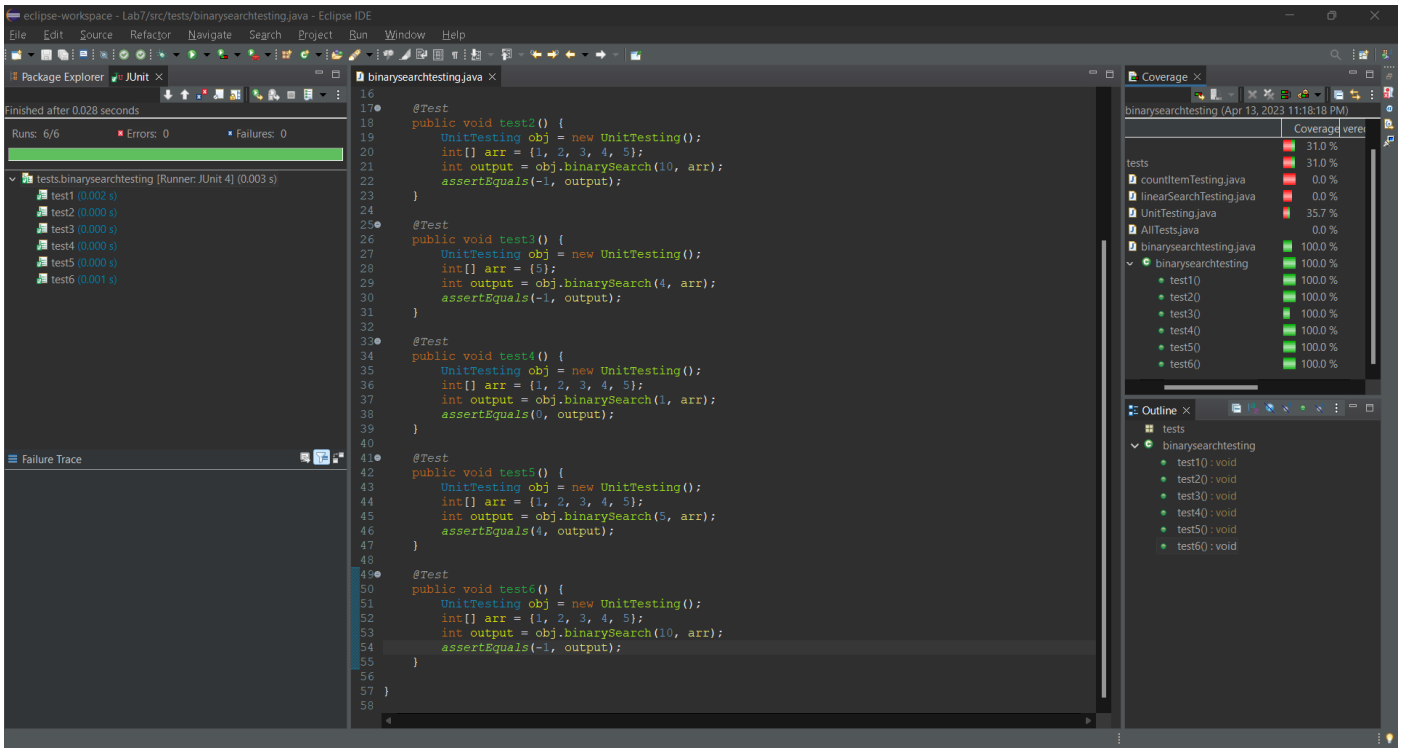
Equivalence Classes:

Input	Expected Output	a	v	Actual Output
v is present in a	Index of v	[1, 2, 3, 4, 2]	2	1
v is not present in a	-1	[1, 2, 3, 4, 5]	10	-1

Boundary Value Analysis:

Input	Expected Output	a	v	Actual Output
Empty array a	-1	[]	4	-1
v is present at the first index of a	0	[1, 2, 3, 4, 5]	1	0
v is present at the last index of the length of a	last index	[1, 2, 3, 4, 5]	5	4
v is not present in a	-1	[1, 2, 3, 4, 5]	10	-1


eclipse-workspace - Lab7/src/tests/binarysearchtesting.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit
Finished after 0.028 seconds
Runs: 6/6 Errors: 0 Failures: 0
tests.binarysearchtesting [Runner: JUnit 4] (0.003 s)
test1 (0.002 s)
test2 (0.000 s)
test3 (0.000 s)
test4 (0.000 s)
test5 (0.000 s)
test6 (0.001 s)
Failure Trace
binarysearchtesting.java
1 package tests;
2
3 import static org.junit.Assert.*;
4
5 public class binarysearchtesting {
6
7 @Test
8 public void test1() {
9 UnitTesting obj = new UnitTesting();
10 int[] arr = {1, 2, 3, 4, 5};
11 int output = obj.binarySearch(2, arr);
12 assertEquals(1, output);
13 }
14
15 @Test
16 public void test2() {
17 UnitTesting obj = new UnitTesting();
18 int[] arr = {1, 2, 3, 4, 5};
19 int output = obj.binarySearch(10, arr);
20 assertEquals(-1, output);
21 }
22
23 @Test
24 public void test3() {
25 UnitTesting obj = new UnitTesting();
26 int[] arr = {5};
27 int output = obj.binarySearch(4, arr);
28 assertEquals(-1, output);
29 }
30
31 @Test
32 public void test4() {
33 UnitTesting obj = new UnitTesting();
34 int[] arr = {1, 2, 3, 4, 5};
35 int output = obj.binarySearch(1, arr);
36 assertEquals(0, output);
37 }
38
39 @Test
40 public void test5() {
41 UnitTesting obj = new UnitTesting();
42 int[] arr = {1, 2, 3, 4, 5};
43 int output = obj.binarySearch(5, arr);
44
45
Coverage
binarysearchtesting (Apr 13, 2023 11:18:18 PM)
Coverage
31.0 %
tests 31.0 %
countItemTesting.java 0.0 %
linearSearchTesting.java 0.0 %
UnitTesting.java 35.7 %
AllTests.java 0.0 %
binarysearchtesting.java 100.0 %
binarysearchtesting 100.0 %
test1() 100.0 %
test2() 100.0 %
test3() 100.0 %
test4() 100.0 %
test5() 100.0 %
test6() 100.0 %
Outline
tests
binarysearchtesting
test1() : void
test2() : void
test3() : void
test4() : void
test5() : void
test6() : void


eclipse-workspace - Lab7/src/tests/binarysearchtesting.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit
Finished after 0.028 seconds
Runs: 6/6 Errors: 0 Failures: 0
tests.binarysearchtesting [Runner: JUnit 4] (0.003 s)
test1 (0.002 s)
test2 (0.000 s)
test3 (0.000 s)
test4 (0.000 s)
test5 (0.000 s)
test6 (0.001 s)
Failure Trace
binarysearchtesting.java
16
17 @Test
18 public void test2() {
19 UnitTesting obj = new UnitTesting();
20 int[] arr = {1, 2, 3, 4, 5};
21 int output = obj.binarySearch(10, arr);
22 assertEquals(-1, output);
23 }
24
25 @Test
26 public void test3() {
27 UnitTesting obj = new UnitTesting();
28 int[] arr = {5};
29 int output = obj.binarySearch(4, arr);
30 assertEquals(-1, output);
31 }
32
33 @Test
34 public void test4() {
35 UnitTesting obj = new UnitTesting();
36 int[] arr = {1, 2, 3, 4, 5};
37 int output = obj.binarySearch(1, arr);
38 assertEquals(0, output);
39 }
40
41 @Test
42 public void test5() {
43 UnitTesting obj = new UnitTesting();
44 int[] arr = {1, 2, 3, 4, 5};
45 int output = obj.binarySearch(5, arr);
46 assertEquals(4, output);
47 }
48
49 @Test
50 public void test6() {
51 UnitTesting obj = new UnitTesting();
52 int[] arr = {1, 2, 3, 4, 5};
53 int output = obj.binarySearch(10, arr);
54 assertEquals(-1, output);
55 }
56
57 }
58
Coverage
binarysearchtesting (Apr 13, 2023 11:18:18 PM)
Coverage
31.0 %
tests 31.0 %
countItemTesting.java 0.0 %
linearSearchTesting.java 0.0 %
UnitTesting.java 35.7 %
AllTests.java 0.0 %
binarysearchtesting.java 100.0 %
binarysearchtesting 100.0 %
test1() 100.0 %
test2() 100.0 %
test3() 100.0 %
test4() 100.0 %
test5() 100.0 %
test6() 100.0 %
Outline
tests
binarysearchtesting
test1() : void
test2() : void
test3() : void
test4() : void
test5() : void
test6() : void

P4. triangle

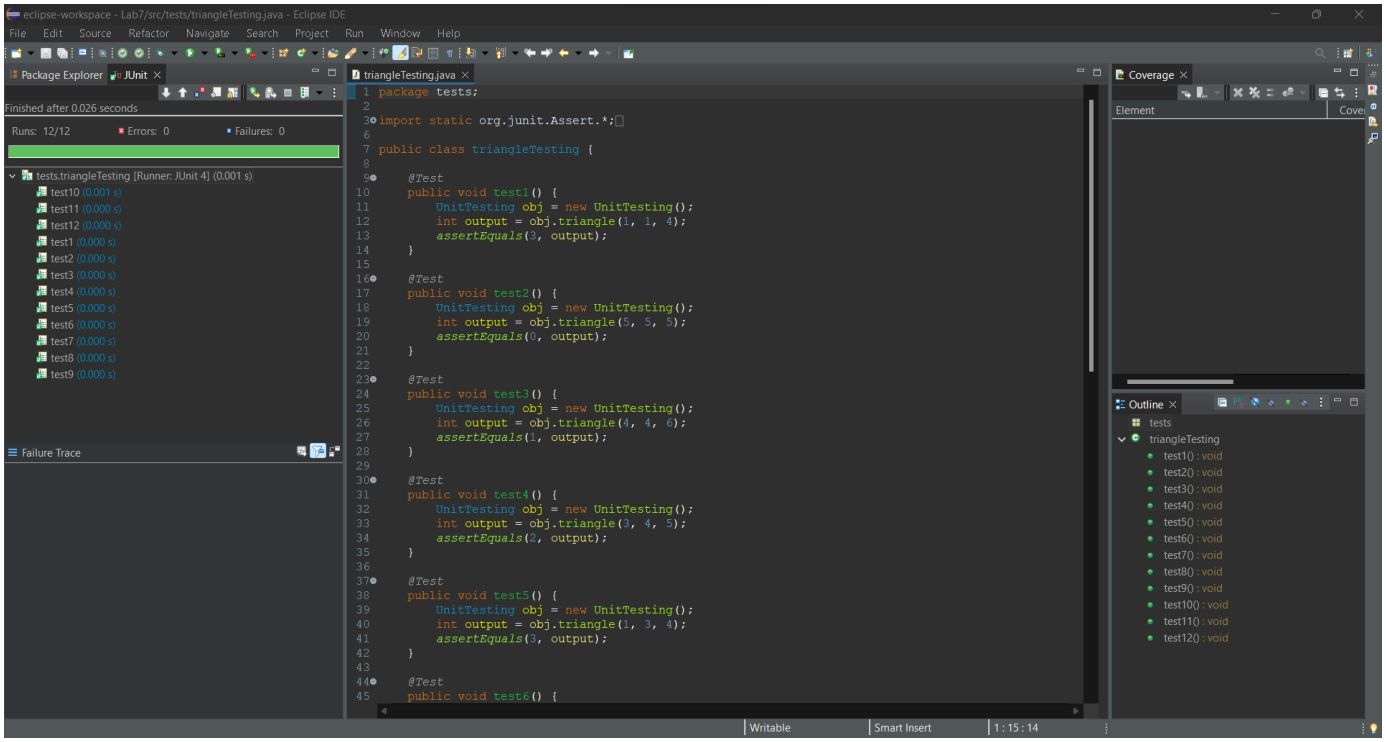
```
final int EQUILATERAL = 0;
final int ISOSCELES = 1;
final int SCALENE = 2;
final int INVALID = 3;
int triangle(int a, int b, int c)
{
      if (a >= b+c || b >= a+c || c >= a+b)
            return(INVALID);
      if (a == b && b == c)
            return(EQUILATERAL);
      if (a == b || a == c || b == c)
            return(ISOSCELES);
      return(SCALENE);


}
```

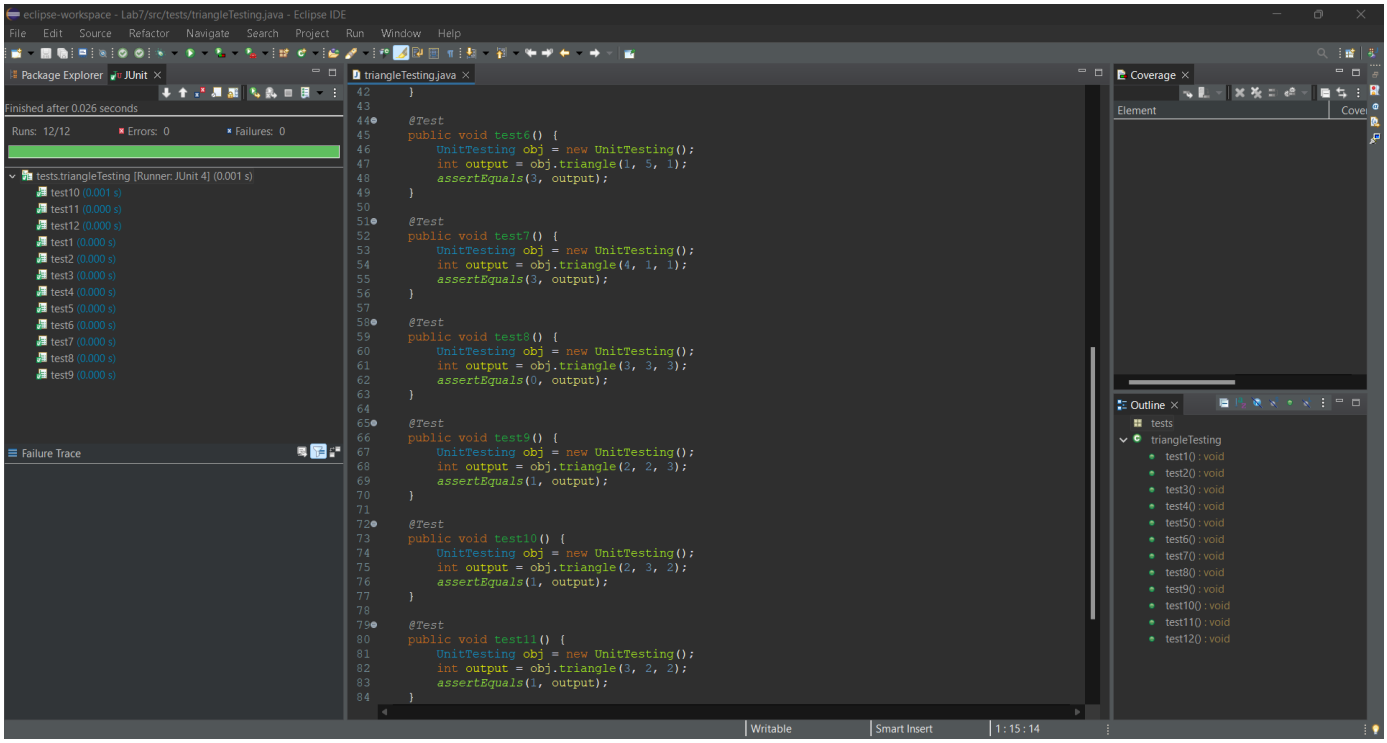
Equivalence Classes:

Input	Expected Output	a	b	c	Actual Output
Invalid triangle (a+b<=c)	INVALID	1	1	4	INVALID
Valid equilateral triangle (a=b=c)	EQUILATERAL	5	5	5	EQUILATERAL
Valid isosceles triangle (a=b<c)	ISOSCELES	4	4	6	ISOSCELES
Valid scalene triangle (a<b<c)	SCALENE	3	4	5	SCALENE

Boundary value Analysis:

Input	Expected Output	a	b	c	Actual Output
Invalid triangle (a+b<=c)	INVALID	1	3	4	INVALID
Invalid triangle (a+c<b)	INVALID	1	5	1	INVALID
Invalid triangle (b+c<a)	INVALID	4	1	1	INVALID
Valid equilateral triangle (a=b=c)	EQUILATERAL	3	3	3	EQUILATERAL
Valid isosceles triangle (a=b<c)	ISOSCELES	2	2	3	ISOSCELES
Valid isosceles triangle (a=c<b)	ISOSCELES	2	3	2	ISOSCELES
Valid isosceles triangle (b=c<a)	ISOSCELES	3	2	2	ISOSCELES
Valid scalene triangle (a<b<c)	SCALENE	3	4	5	SCALENE


eclipse-workspace - Lab7/src/tests/triangleTesting.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit
Finished after 0.026 seconds
Runs: 12/12 Errors: 0 Failures: 0
tests.triangleTesting [Runner: JUnit 4] (0.001 s)
test10 (0.001 s)
test11 (0.000 s)
test12 (0.000 s)
test1 (0.000 s)
test2 (0.000 s)
test3 (0.000 s)
test4 (0.000 s)
test5 (0.000 s)
test6 (0.000 s)
test7 (0.000 s)
test8 (0.000 s)
test9 (0.000 s)
Failure Trace
triangleTesting.java
1 package tests;
3 import static org.junit.Assert.*;
7 public class triangleTesting {
9 @Test
10 public void test1() {
11 UnitTesting obj = new UnitTesting();
12 int output = obj.triangle(1, 1, 4);
13 assertEquals(3, output);
14 }
16 @Test
17 public void test2() {
18 UnitTesting obj = new UnitTesting();
19 int output = obj.triangle(5, 5, 5);
20 assertEquals(0, output);
21 }
23 @Test
24 public void test3() {
25 UnitTesting obj = new UnitTesting();
26 int output = obj.triangle(4, 4, 6);
27 assertEquals(1, output);
28 }
30 @Test
31 public void test4() {
32 UnitTesting obj = new UnitTesting();
33 int output = obj.triangle(3, 4, 5);
34 assertEquals(2, output);
35 }
37 @Test
38 public void test5() {
39 UnitTesting obj = new UnitTesting();
40 int output = obj.triangle(1, 3, 4);
41 assertEquals(3, output);
42 }
44 @Test
45 public void test6() {
Coverage
Element
Outline
tests
triangleTesting
test1() : void
test2() : void
test3() : void
test4() : void
test5() : void
test6() : void
test7() : void
test8() : void
test9() : void
test10() : void
test11() : void
test12() : void
Writable
Smart Insert
1 : 15 : 14


eclipse-workspace - Lab7/src/tests/triangleTesting.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit
Finished after 0.026 seconds
Runs: 12/12 Errors: 0 Failures: 0
tests.triangleTesting [Runner: JUnit 4] (0.001 s)
test10 (0.001 s)
test11 (0.000 s)
test12 (0.000 s)
test1 (0.000 s)
test2 (0.000 s)
test3 (0.000 s)
test4 (0.000 s)
test5 (0.000 s)
test6 (0.000 s)
test7 (0.000 s)
test8 (0.000 s)
test9 (0.000 s)
Failure Trace
triangleTesting.java
42 }
44 @Test
45 public void test6() {
46 UnitTesting obj = new UnitTesting();
47 int output = obj.triangle(1, 5, 1);
48 assertEquals(3, output);
49 }
51 @Test
52 public void test7() {
53 UnitTesting obj = new UnitTesting();
54 int output = obj.triangle(4, 1, 1);
55 assertEquals(3, output);
56 }
58 @Test
59 public void test8() {
60 UnitTesting obj = new UnitTesting();
61 int output = obj.triangle(3, 3, 3);
62 assertEquals(0, output);
63 }
65 @Test
66 public void test9() {
67 UnitTesting obj = new UnitTesting();
68 int output = obj.triangle(2, 2, 3);
69 assertEquals(1, output);
70 }
72 @Test
73 public void test10() {
74 UnitTesting obj = new UnitTesting();
75 int output = obj.triangle(2, 3, 2);
76 assertEquals(1, output);
77 }
75 @Test
80 public void test11() {
81 UnitTesting obj = new UnitTesting();
82 int output = obj.triangle(3, 2, 2);
83 assertEquals(1, output);
84 }
Coverage
Element
Outline
tests
triangleTesting
test1() : void
test2() : void
test3() : void
test4() : void
test5() : void
test6() : void
test7() : void
test8() : void
test9() : void
test10() : void
test11() : void
test12() : void
Writable
Smart Insert
1 : 15 : 14

P5. prefix

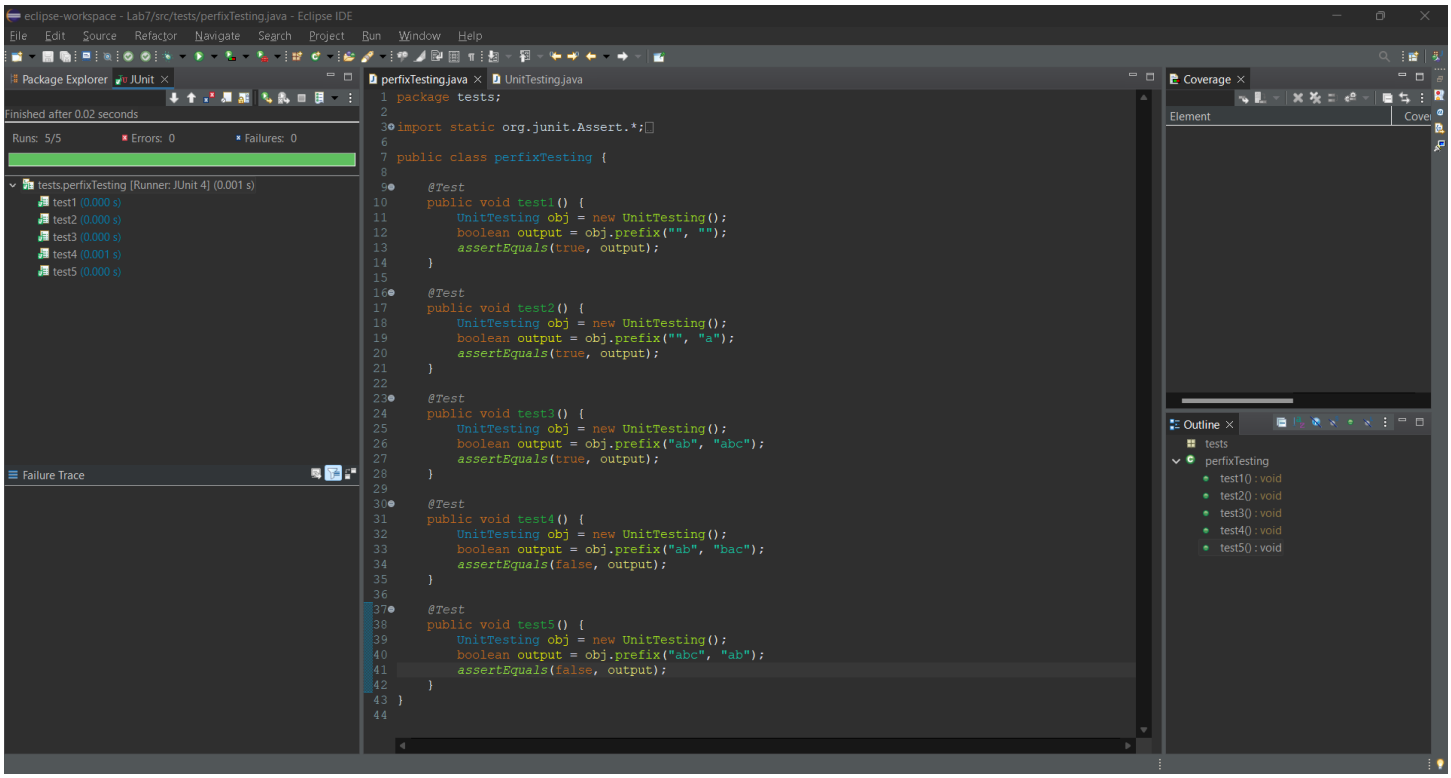
```
public static boolean prefix(String s1, String s2)
{
      if (s1.length() > s2.length())
      {
            return false;
      }
      for (int i = 0; i < s1.length(); i++)
      {
            if (s1.charAt(i) != s2.charAt(i))
            {
                  return false;
            }
      }
      return true;
}
```

Equivalence Classes:

Input	**Expected Output**	s1	s2	**Actual Output**
Empty string s1 and s2	True	“”	“”	True
Empty string s1 and non-empty s2	True	“”	“a”	True
Non-empty s1 is a prefix of non-empty s2	True	“ab”	“abc”	True
Non-empty s1 is not a prefix of s2	False	“ab”	“bac”	False
Non-empty s1 is longer than s2	False	“abc”	“ab”	False

Boundary Value Analysis:

Input	Expected Output	s1	s2	Actual Output
Empty string s1 and s2	True	“”	“”	True
Empty string s1 and non-empty s2	True	“”	“a”	True
Non-empty s1 is not a prefix of s2	False	“ab”	“bac”	False
Non-empty s1 is longer than s2	False	“abc”	“ab”	False



P6. prefix

a) Identify the equivalence classes for the system

Equivalence Classes:

EC1: All sides are positive, real numbers.

EC2: One or more sides are negative or zero.

EC3: The sum of the lengths of any two sides is less than or equal to the length of the remaining side (impossible lengths).

EC4: The sum of the lengths of any two sides is greater than the length of the remaining side (possible lengths).

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

Test cases:

TC1 (EC1): A=3, B=4, C=5 (right-angled triangle)

TC2 (EC1): A=5, B=5, C=5 (equilateral triangle)

TC3 (EC1): A=5, B=6, C=7 (scalene triangle)

TC4 (EC1): A=5, B=5, C=7 (isosceles triangle)

TC5 (EC2): A=-2, B=4, C=5 (invalid input)

TC6 (EC2): A=0, B=4, C=5 (invalid input)

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

Test cases for the boundary condition A + B > C:

TC7 (EC4): A=2, B=3, C=6 (sum of A and B is equal to C)

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

Test cases for the boundary condition A = C:

TC8 (EC4): A=5, B=6, C=5 (A equals to C)

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

Test cases for the boundary condition A = B = C:

TC9 (EC4): A=1, B=1, C=1 (all sides are equal)

f) For the boundary condition A2 + B2 = C2 case (right-angle triangle), identify test cases to verify the boundary.

Test cases for the boundary condition A^2 + B^2 = C^2:

TC10 (EC4): A=3, B=4, C=5 (right-angled triangle)

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

Test cases for the non-triangle case:

TC11 (EC3): A=2, B=2, C=4 (sum of A and B is less than C)

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

Test points for non-positive input:

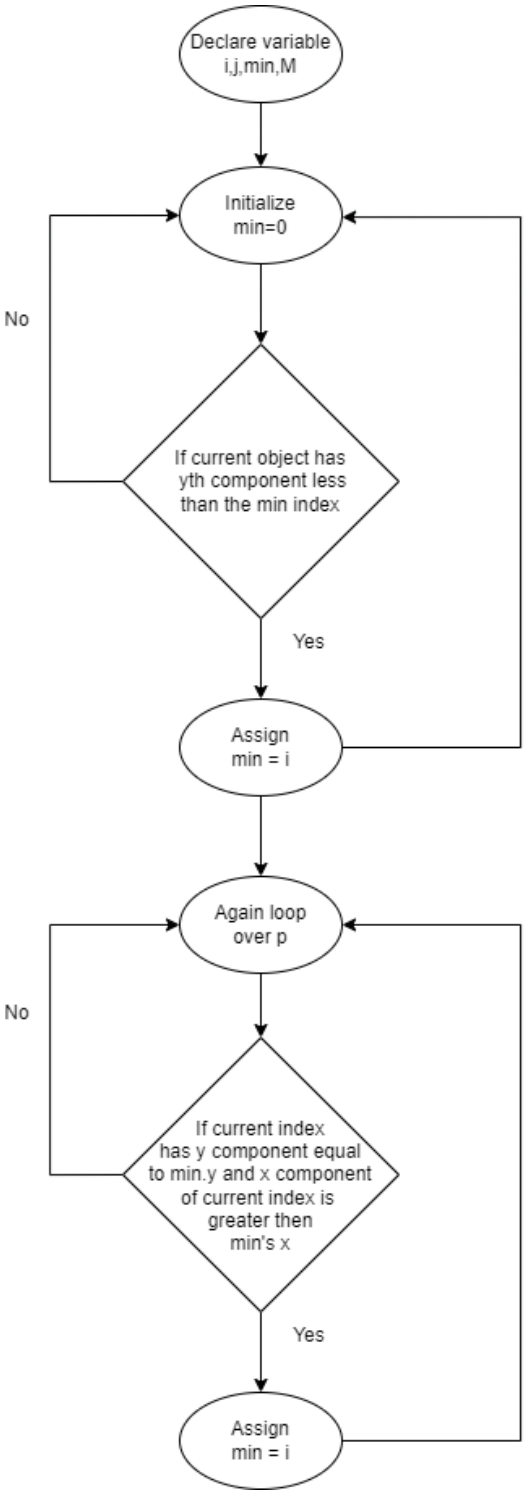
TP1 (EC2): A=0, B=4, C=5 (invalid input)

TP2 (EC2): A=-2, B=4, C=5 (invalid input)

Note: Test cases TC1 to TC10 covers all identified equivalence classes.

Section B:

(1) Control flow graph



(2) Test Cases

(a) Statement coverage test set: In this, all the statements in the code should be covered

Test Number	Test Case
1	p is an empty array
2	p has one point object
3	p has two points object with different y component
4	p has two points object with different x component
5	p has three or more point object with different y component

(b) Branch Coverage test set: In this, all branches are taken at least once

Test Number	Test Case
1	p is an empty array
2	p has one point object
3	p has two points object with different y component
4	p has two points object with different x component
5	p has three or more point object with different y component
6	p has three or more point object with same y component
7	p has three or more point object with all same x component
8	p has three or more point object with all different x component

9	p has three or more point object with some same and some different x component

(b) Basic condition coverage test set: Each boolean expression has been evaluated to be both true and false

Test Number	Test Case
1	p is an empty array
2	p has one point object
3	p has two points object with different y component
4	p has two points object with different x component
5	p has three or more point objects with different y component
6	p has three or more point objects with same y component
7	p has three or more point objects with all same x component
8	p has three or more point objects with all different x component
9	p has three or more point objects with some same and some different x component
10	p has three or more point objects with some same and some different y component
11	p has three or more point objects with all different y component
12	p has three or more point objects with all same y component