LAB 7

Name: - Aashka Arvindbhai Thumar

Student ID: - 202001205

Section: - B

Lab Group: - 4

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?**

Input	Day	Month	Year	Expected Output
Case 1	15	4	2011	14/4/2011
Case 2	15	4	1907	14/4/1907
Case 3	1	6	2012	31/5/2012
Case 4	29	2	2007	Invalid
Case 5	29	2	2008	28/2/2008
Case 6	31	2	2004	Invalid
Case 7	30	3	2007	29/3/2007
Case 8	2	7	2021	1/7/2014
Case 9	1	1	1990	Invalid
Case 10	30	2	2020	Invalid

Equivalence Class Partition

Day:

Partition ID	Range	Status
E1	Between 1 and 28	Valid
E2	Less than 1	Invalid
E3	Greater than 31	Invalid
E4	Equals 30	Valid
E5	Equals 29	Valid for leap year
E6	Equals 31	Valid

Month:

Partition ID	Range	Status
E7	Between 1 and 12	Valid
E8	Less than 1	Invalid
E9	Greater than 12	Invalid

Year:

Partition ID	Range	Status
E10	Between 1990 and 2015	Valid
E11	Less than 1	Invalid
E12	Greater than 2015	Invalid

Programs:

P1. The function linearSearch 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.

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

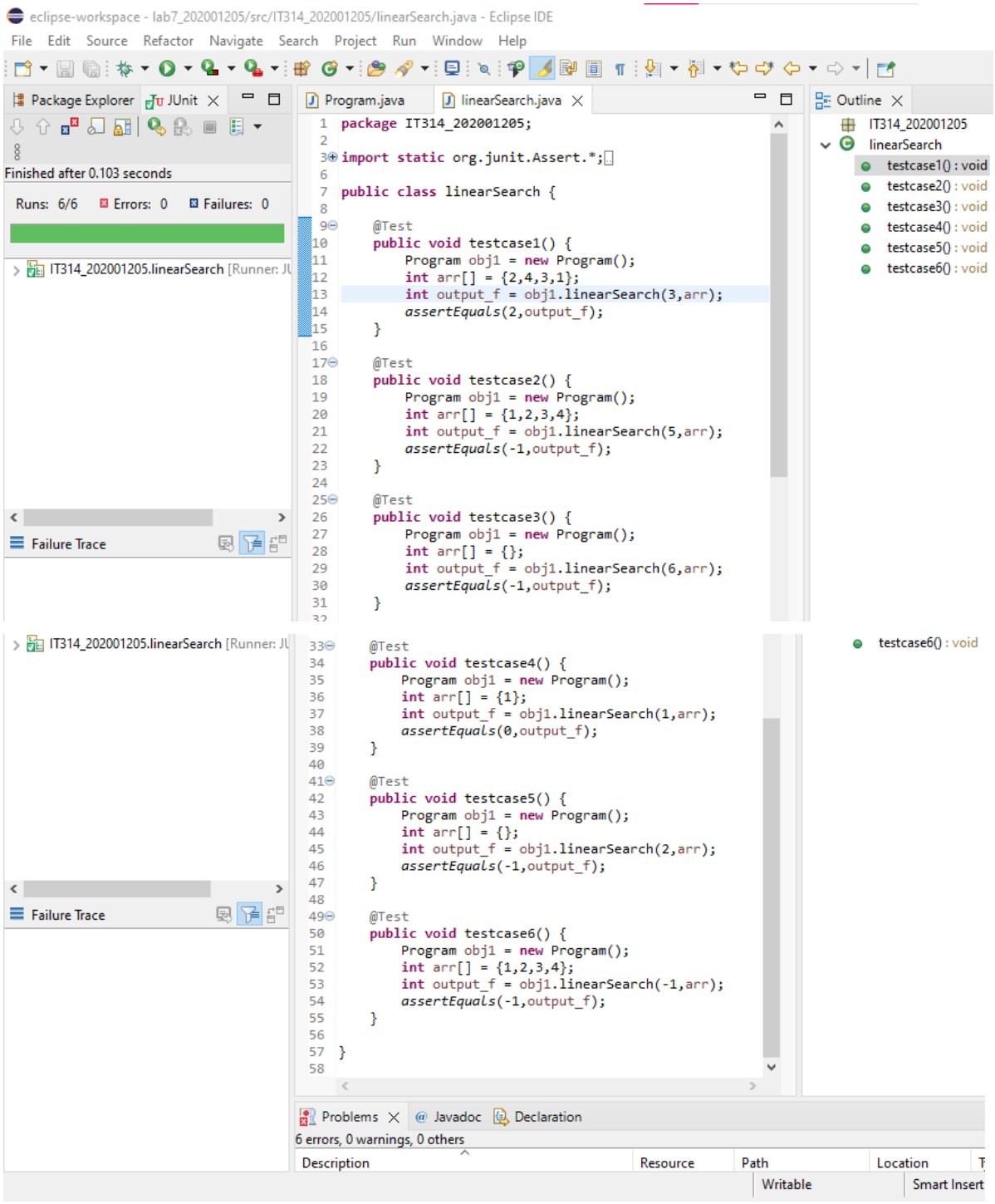
Test Cases:

Equivalence Class Testing

Test Cases	Target	Array	Output	Expected Output
1	3	{2, 4, 3, 1}	2	2
2	5	{1, 2, 3, 4}	-1	-1
3	6	{ }	-1	-1

Boundary Value Analysis

Test Cases	Target	Array	Output	Expected Output
1	1	{1}	0	0
2	2	{ }	-1	-1
3	-1	{1,2,3,4}	-1	-1


eclipse-workspace - lab7_202001205/src/IT314_202001205/linearSearch.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
JUnit
Finished after 0.103 seconds
Runs: 6/6
Errors: 0
Failures: 0
IT314_202001205.linearSearch [Runner: JU
Failure Trace
Program.java
linearSearch.java
1 package IT314_202001205;
2
3 import static org.junit.Assert.*;
6
7 public class linearSearch {
8
9 @Test
10 public void testcase1() {
11 Program obj1 = new Program();
12 int arr[] = {2,4,3,1};
13 int output_f = obj1.linearSearch(3,arr);
14 assertEquals(2,output_f);
15 }
16
17 @Test
18 public void testcase2() {
19 Program obj1 = new Program();
20 int arr[] = {1,2,3,4};
21 int output_f = obj1.linearSearch(5,arr);
22 assertEquals(-1,output_f);
23 }
24
25 @Test
26 public void testcase3() {
27 Program obj1 = new Program();
28 int arr[] = {};
29 int output_f = obj1.linearSearch(6,arr);
30 assertEquals(-1,output_f);
31 }
32
33 @Test
34 public void testcase4() {
35 Program obj1 = new Program();
36 int arr[] = {1};
37 int output_f = obj1.linearSearch(1,arr);
38 assertEquals(0,output_f);
39 }
40
41 @Test
42 public void testcase5() {
43 Program obj1 = new Program();
44 int arr[] = {};
45 int output_f = obj1.linearSearch(2,arr);
46 assertEquals(-1,output_f);
47 }
48
49 @Test
50 public void testcase6() {
51 Program obj1 = new Program();
52 int arr[] = {1,2,3,4};
53 int output_f = obj1.linearSearch(-1,arr);
54 assertEquals(-1,output_f);
55 }
56
57 }
58
Outline
IT314_202001205
linearSearch
testcase1() : void
testcase2() : void
testcase3() : void
testcase4() : void
testcase5() : void
testcase6() : void
Problems
Javadoc
Declaration
6 errors, 0 warnings, 0 others
Description
Resource
Path
Location
Writable
Smart Insert

P2. The function countItem returns the number of times a value v appears in an array of integers a.

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

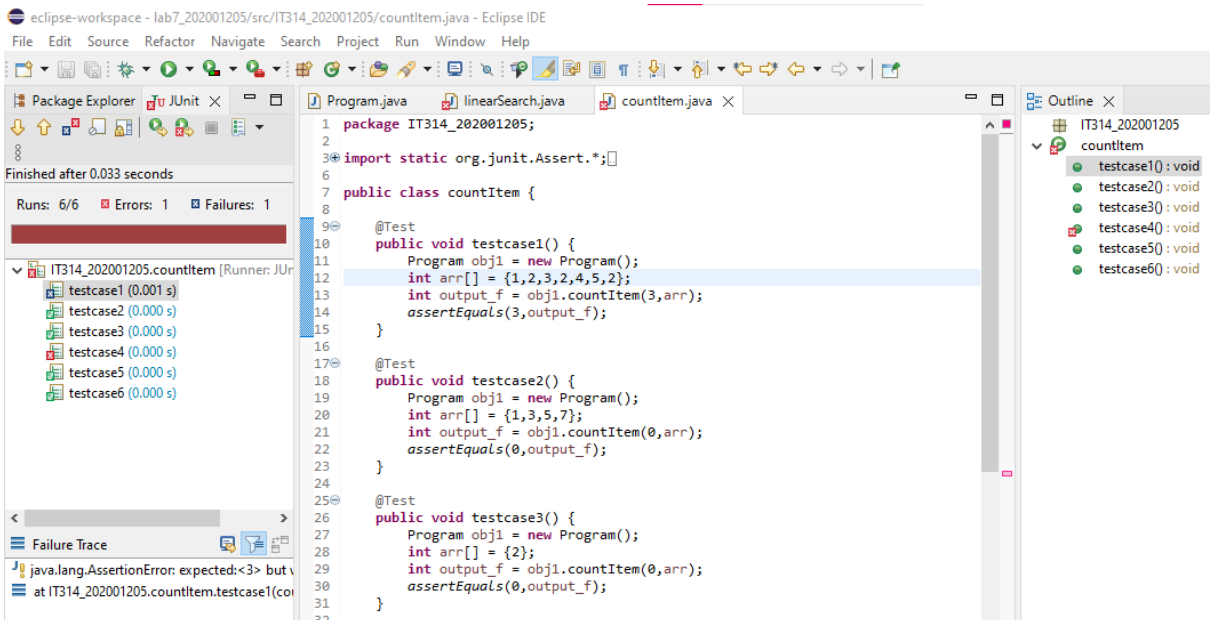
Test cases:

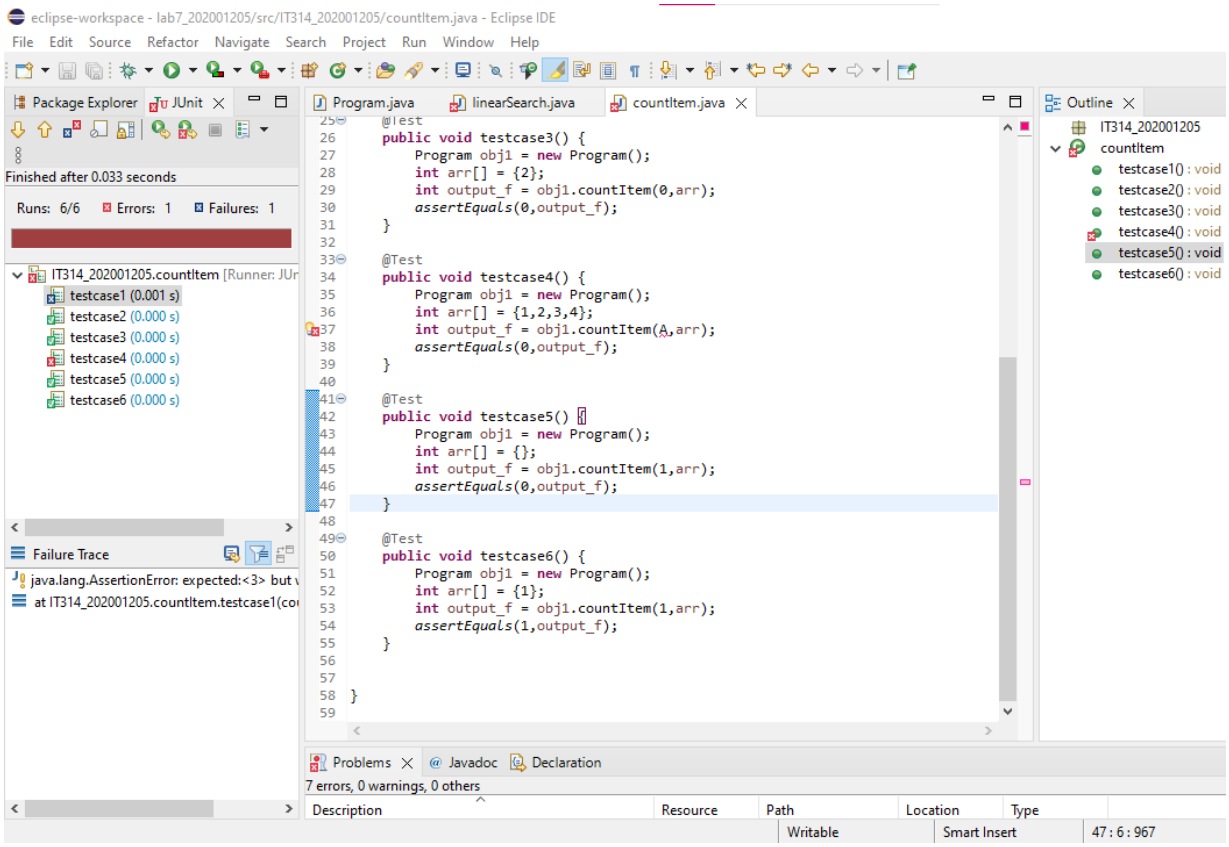
Equivalence Class Testing

Test Cases	Target	Array	Output	Expected Output
1	2	{ 1,2,3,2,4,5,2};	3	3
2	2	{ 1,3,5,7};	0	0
3	1	{2};	0	0
4	A	{1,2,3,4};	Error	0

Boundary Value Analysis

Test Cases	Array	Target	Output	Expected Output
1	{};	1	0	0
2	{1};	1	1	1


eclipse-workspace - lab7_202001205/src/IT314_202001205/countItem.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit
Finished after 0.033 seconds
Runs: 6/6 Errors: 1 Failures: 1
IT314_202001205.countItem [Runner: JUr
testcase1 (0.001 s)
testcase2 (0.000 s)
testcase3 (0.000 s)
testcase4 (0.000 s)
testcase5 (0.000 s)
testcase6 (0.000 s)
Failure Trace
java.lang.AssertionError: expected:<3> but
at IT314_202001205.countItem.testcase1(co
Program.java linearSearch.java countItem.java
1 package IT314_202001205;
2
3 import static org.junit.Assert.*;
6
7 public class countItem {
8
9 @Test
10 public void testcase1() {
11 Program obj1 = new Program();
12 int arr[] = {1,2,3,2,4,5,2};
13 int output_f = obj1.countItem(3,arr);
14 assertEquals(3,output_f);
15 }
16
17 @Test
18 public void testcase2() {
19 Program obj1 = new Program();
20 int arr[] = {1,3,5,7};
21 int output_f = obj1.countItem(0,arr);
22 assertEquals(0,output_f);
23 }
24
25 @Test
26 public void testcase3() {
27 Program obj1 = new Program();
28 int arr[] = {2};
29 int output_f = obj1.countItem(0,arr);
30 assertEquals(0,output_f);
31 }
Outline
IT314_202001205
countItem
testcase1() : void
testcase2() : void
testcase3() : void
testcase4() : void
testcase5() : void
testcase6() : void


eclipse-workspace - lab7_202001205/src/IT314_202001205/countItem.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit
Finished after 0.033 seconds
Runs: 6/6 Errors: 1 Failures: 1
IT314_202001205.countItem [Runner: JUr
testcase1 (0.001 s)
testcase2 (0.000 s)
testcase3 (0.000 s)
testcase4 (0.000 s)
testcase5 (0.000 s)
testcase6 (0.000 s)
Failure Trace
java.lang.AssertionError: expected:<3> but
at IT314_202001205.countItem.testcase1(co
Program.java linearSearch.java countItem.java
@Test
public void testcase3() {
Program obj1 = new Program();
int arr[] = {2};
int output_f = obj1.countItem(0,arr);
assertEquals(0,output_f);
}
@Test
public void testcase4() {
Program obj1 = new Program();
int arr[] = {1,2,3,4};
int output_f = obj1.countItem(A,arr);
assertEquals(0,output_f);
}
@Test
public void testcase5() {
Program obj1 = new Program();
int arr[] = {};
int output_f = obj1.countItem(1,arr);
assertEquals(0,output_f);
}
@Test
public void testcase6() {
Program obj1 = new Program();
int arr[] = {1};
int output_f = obj1.countItem(1,arr);
assertEquals(1,output_f);
}
}
Outline
IT314_202001205
countItem
testcase1() : void
testcase2() : void
testcase3() : void
testcase4() : void
testcase5() : void
testcase6() : void
Problems Javadoc Declaration
7 errors, 0 warnings, 0 others
Description Resource Path Location Type
Writable Smart Insert 47 : 6 : 967

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.

```
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);
}
```

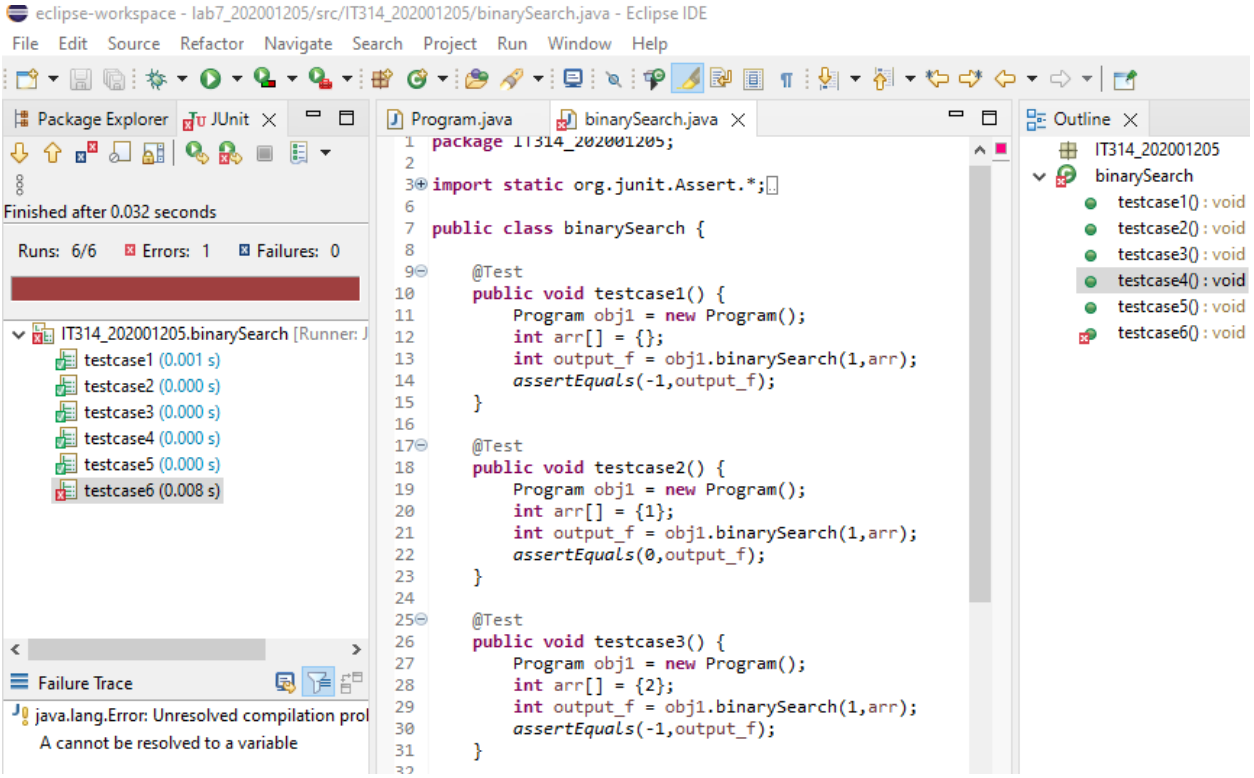
Test cases:

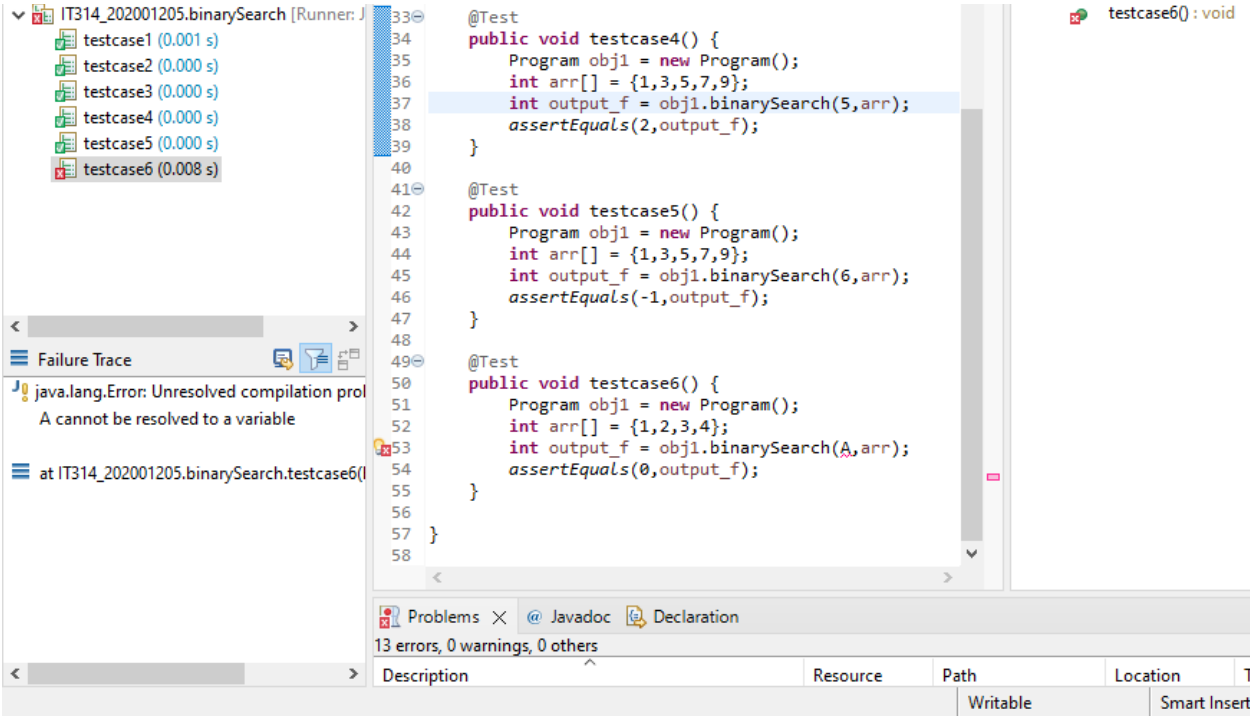
Boundary Class Partitioning

Test Cases	Array	Target	Output	Expected Output
1	{};	1	-1	-1
2	{ 1};	1	0	0
3	{2};	1	-1	-1

Equivalence Class Testing

Test Cases	Array	Target	Output	Expected Output
1	{ 1,3,5,7,9};	5	2	2
2	{ 1,3,5,7,9};	6	-1	-1
3	{1,2,3,4};	A	Error	0




IT314_202001205.binarySearch [Runner: J
testcase1 (0.001 s)
testcase2 (0.000 s)
testcase3 (0.000 s)
testcase4 (0.000 s)
testcase5 (0.000 s)
testcase6 (0.008 s)
Failure Trace
java.lang.Error: Unresolved compilation prol
A cannot be resolved to a variable
at IT314_202001205.binarySearch.testcase6(
33 @Test
34 public void testcase4() {
35 Program obj1 = new Program();
36 int arr[] = {1,3,5,7,9};
37 int output_f = obj1.binarySearch(5,arr);
38 assertEquals(2,output_f);
39 }
40
41 @Test
42 public void testcase5() {
43 Program obj1 = new Program();
44 int arr[] = {1,3,5,7,9};
45 int output_f = obj1.binarySearch(6,arr);
46 assertEquals(-1,output_f);
47 }
48
49 @Test
50 public void testcase6() {
51 Program obj1 = new Program();
52 int arr[] = {1,2,3,4};
53 int output_f = obj1.binarySearch(A,arr);
54 assertEquals(0,output_f);
55 }
56
57 }
58
testcase6() : void
Problems
Javadoc
Declaration
13 errors, 0 warnings, 0 others
Description
Resource
Path
Location
Writable
Smart Insert

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).

```
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);
}
```

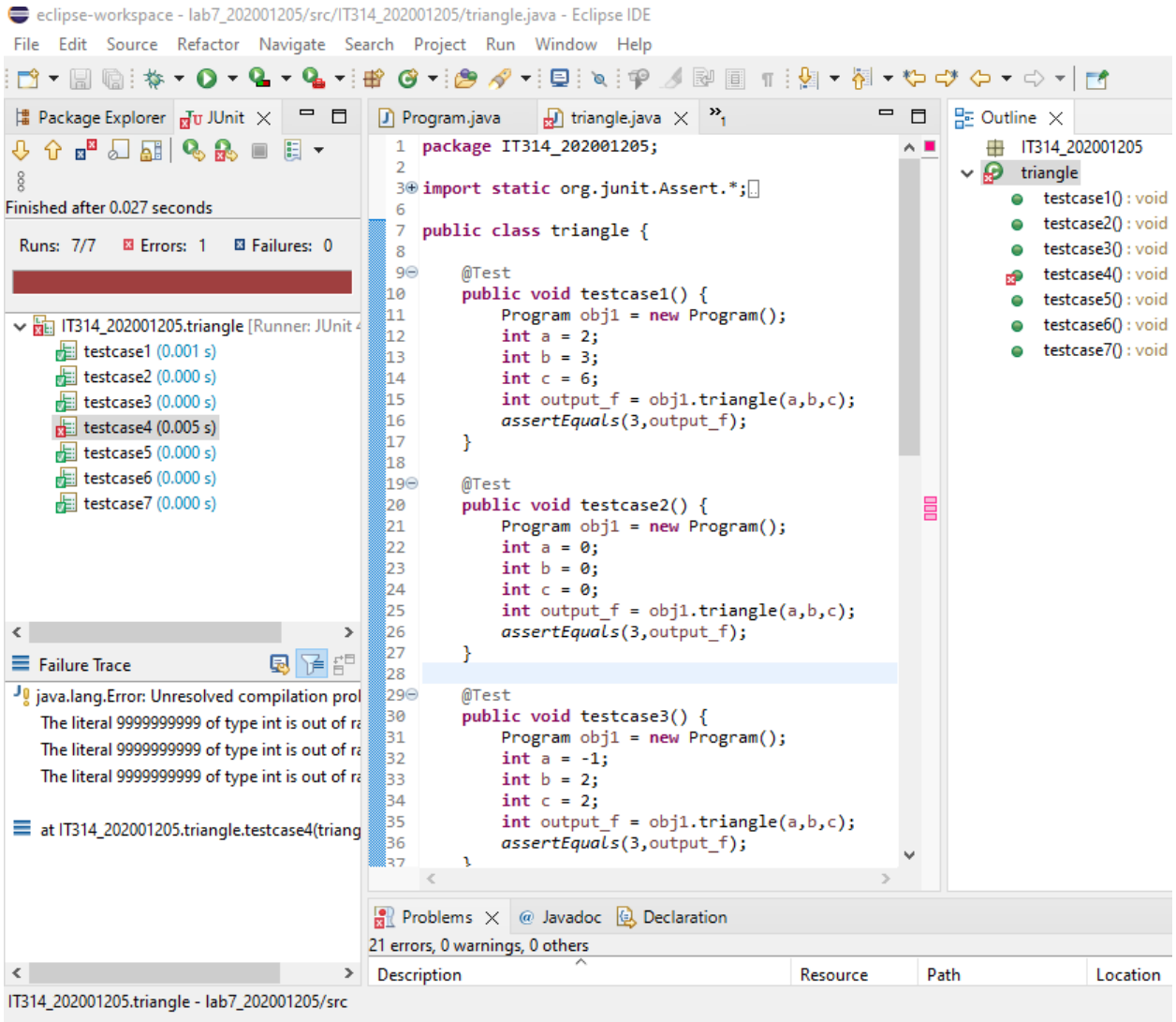
Test cases:

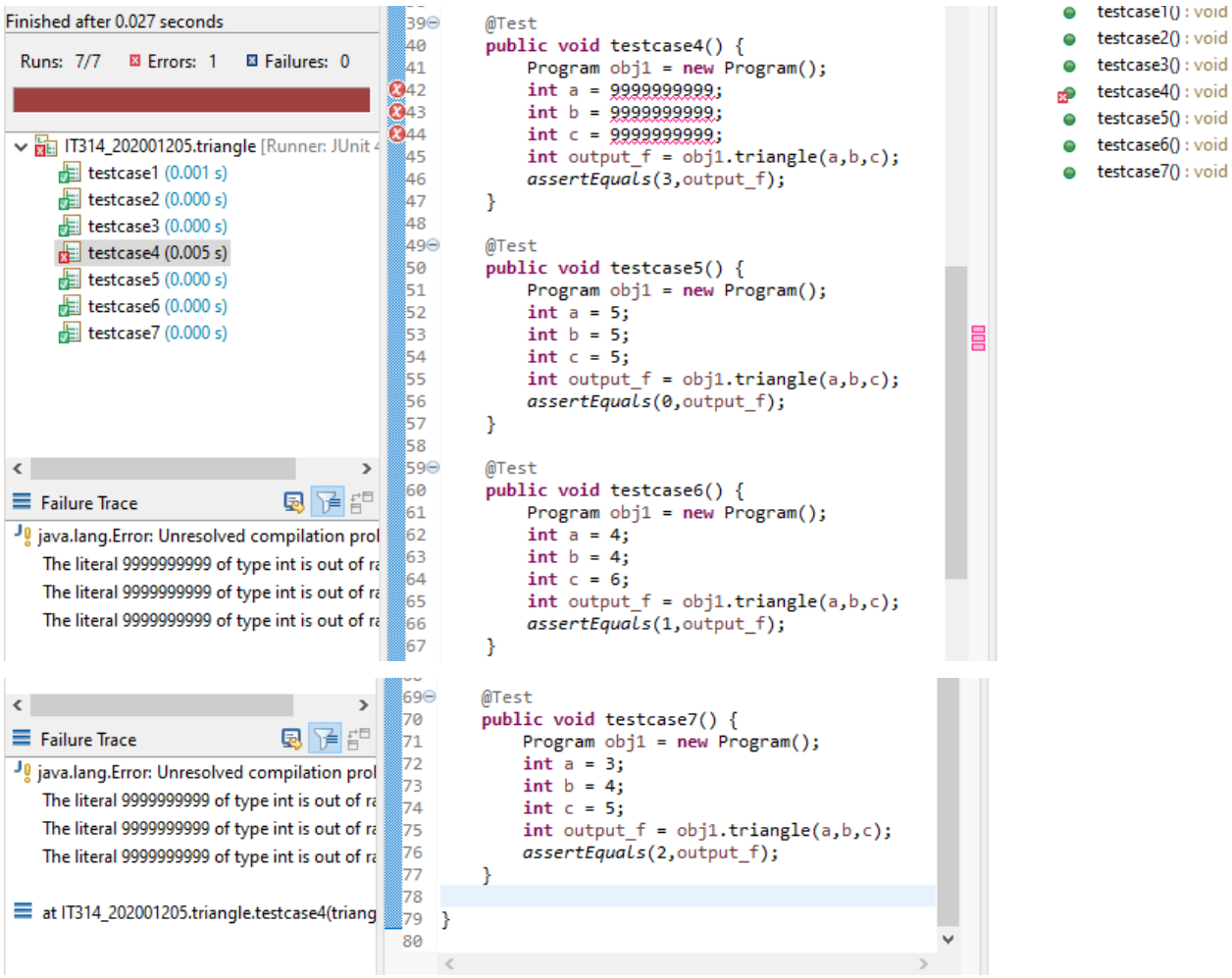
Boundary Class Partitioning

Test Cases	a	b	c	Output	Expected Outcome
1	2	3	6	3	3
2	0	0	0	3	3
3	-1	2	2	3	3
4	9999999999	9999999999	9999999999	Error	3

Equivalence Class Partitioning

Test Cases	a	b	c	Output	Expected Outcome
1	5	5	5	0	0
2	4	4	6	1	1
3	3	4	5	2	2





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;
}
```

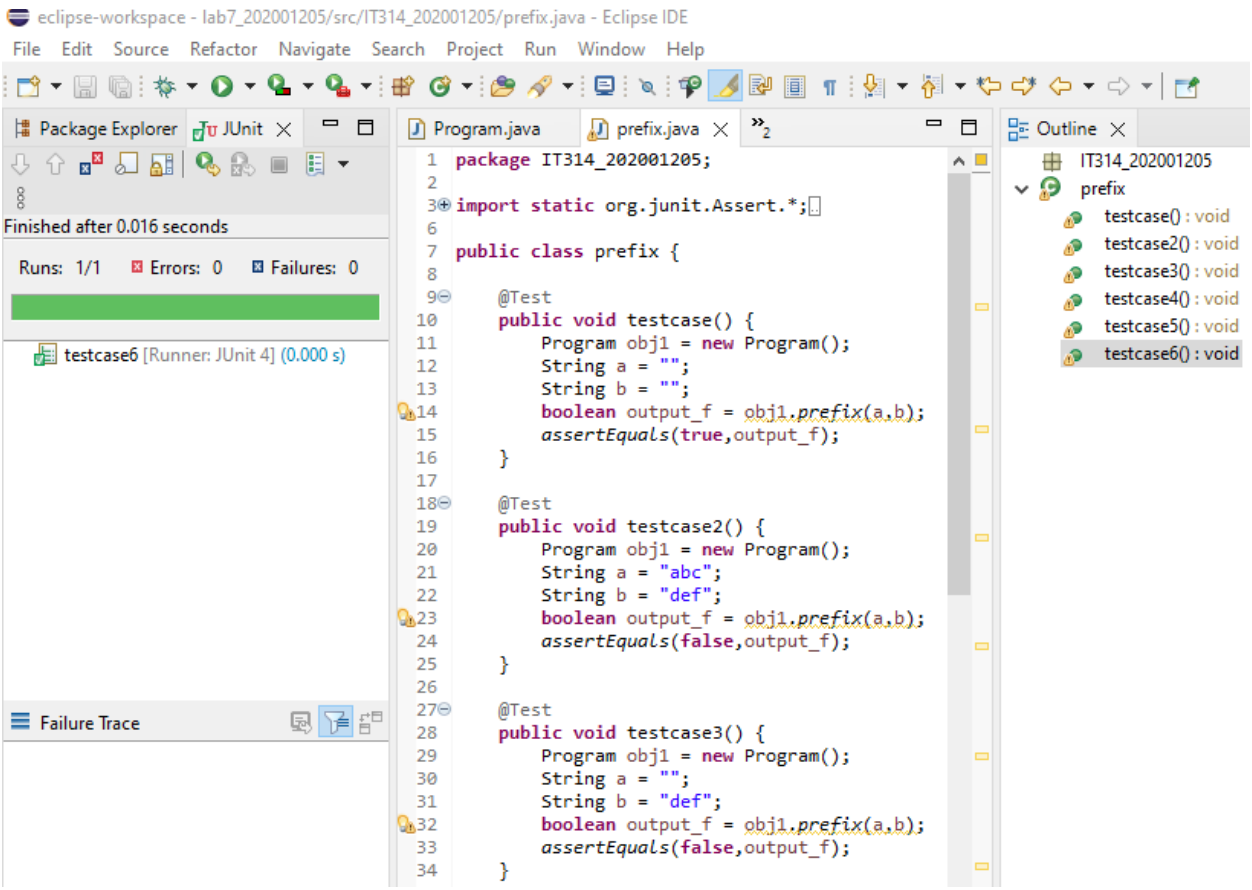
Test Cases:

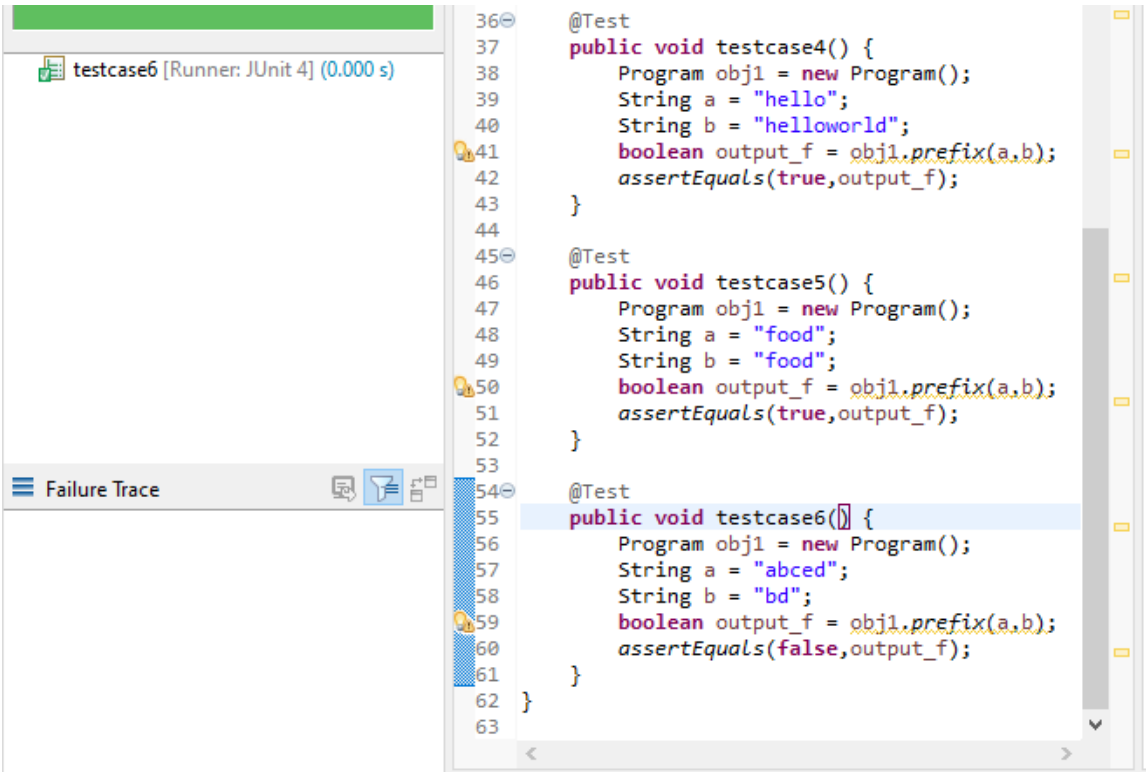
Boundary Class Partitioning

Test Cases	String 1	String 2	Output	Expected Output
1	“”	“”	true	true
2	abc	def	false	false
3	“”	def	true	true

Equivalence Class Partitioning

Test Cases	String 1	String 2	Output	Expected Output
1	hello	helloworld	true	true
2	food	food	true	true
3	abced	bd	false	false


eclipse-workspace - lab7_202001205/src/IT314_202001205/prefix.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
JUnit
Finished after 0.016 seconds
Runs: 1/1
Errors: 0
Failures: 0
testcase6 [Runner: JUnit 4] (0.000 s)
Failure Trace
Program.java
prefix.java
Outline
IT314_202001205
prefix
testcase() : void
testcase2() : void
testcase3() : void
testcase4() : void
testcase5() : void
testcase6() : void
1 package IT314_202001205;
3 import static org.junit.Assert.*;
7 public class prefix {
9 @Test
10 public void testcase() {
11 Program obj1 = new Program();
12 String a = "";
13 String b = "";
14 boolean output_f = obj1.prefix(a,b);
15 assertEquals(true,output_f);
16 }
18 @Test
19 public void testcase2() {
20 Program obj1 = new Program();
21 String a = "abc";
22 String b = "def";
23 boolean output_f = obj1.prefix(a,b);
24 assertEquals(false,output_f);
25 }
27 @Test
28 public void testcase3() {
29 Program obj1 = new Program();
30 String a = "";
31 String b = "def";
32 boolean output_f = obj1.prefix(a,b);
33 assertEquals(false,output_f);
34 }


testcase6 [Runner: JUnit 4] (0.000 s)
Failure Trace
36 @Test
37 public void testcase4() {
38 Program obj1 = new Program();
39 String a = "hello";
40 String b = "helloworld";
41 boolean output_f = obj1.prefix(a,b);
42 assertEquals(true,output_f);
43 }
45 @Test
46 public void testcase5() {
47 Program obj1 = new Program();
48 String a = "food";
49 String b = "food";
50 boolean output_f = obj1.prefix(a,b);
51 assertEquals(true,output_f);
52 }
54 @Test
55 public void testcase6() {
56 Program obj1 = new Program();
57 String a = "abced";
58 String b = "bd";
59 boolean output_f = obj1.prefix(a,b);
60 assertEquals(false,output_f);
61 }
62 }

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

Class ID	Class
E1	All sides are positive
E2	Two of its sides are negative
E3	One of its side is negative
E4	Sum of two sided is less than the third side
E5	Any of the side/sides is negative

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 ID	Class ID	Test Case
T1	E1	A=1, B=1, C=1
T2	E2	A=3, B=4, C=5
T3	E2	A=0, B=0, C=1
T4	E3	A=0, B=1, C=2
T5	E4	A=1, B=3, C=8
T6	E5	A=-1, B=1, C=5

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

A=1, B=3, C=2

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

A=3, B=2, C=3

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

A=30, B=30, C=30

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

A=6, B=8, C=10

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

A=20, B=10, C=5

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

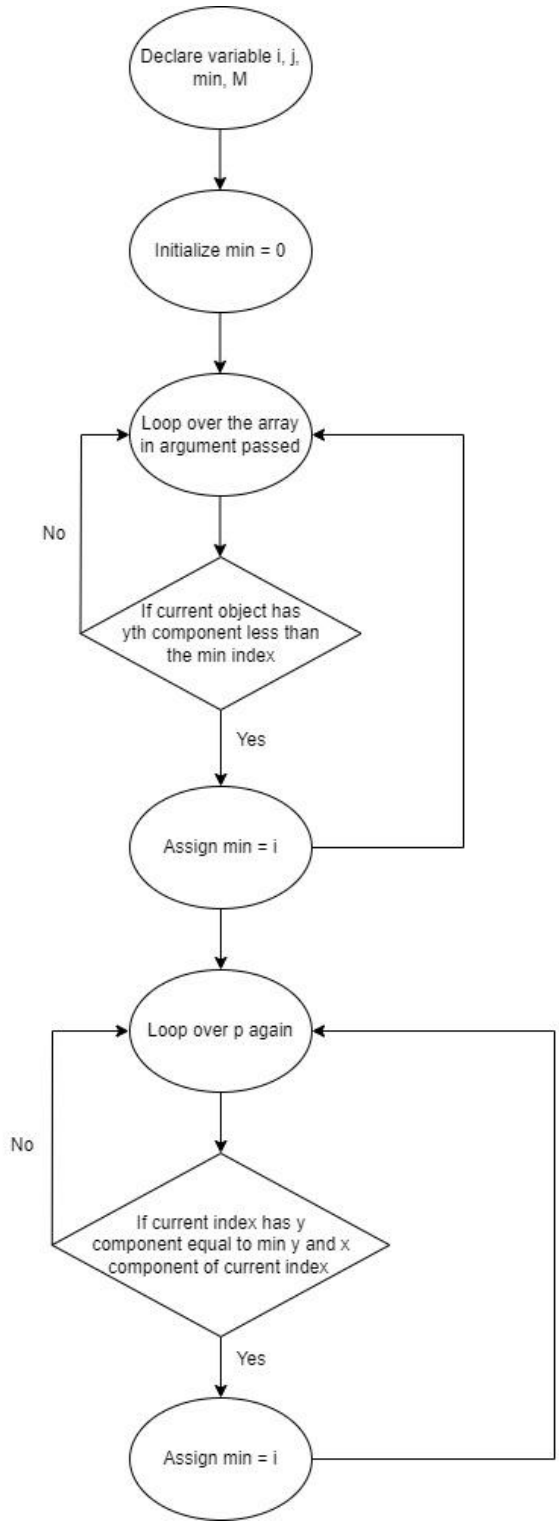
A=0, B=10, C=0

A=0, B=0, C=0

A=0, B=-1, C=10

Section B

1. Control Flow Graph:



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

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

    b. **Branch Coverage**

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 point objects with different y component
4	p has two point objects 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

c. Basic Condition Coverage.

Test Number	Test Case
1	p is an empty array
2	p has one point object
3	p has two point objects with different y component
4	p has two point objects 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

Each boolean expression has been evaluated to both true and false