Name:Aditya Raj
ID:202001099
Lab 7

Section A

Based on the input ranges, we can identify the following equivalence classes:

1. Valid dates: The input triple (day, month, year) that represents a valid date in the Gregorian calendar, such as (3, 4, 1995).
2. Invalid dates: The input triple (day, month, year) that represents an invalid date, such as (31, 2, 2022) or (29, 2, 1900).
3. Out of range dates: The input triple (day, month, year) that are outside the allowed ranges, such as (0, 5, 2010) or (15, 13, 2005). Based on these equivalence classes, we can design the following test cases:

Tester Action and Input Data Expected Outcome

Valid dates:

1. Calculate previous date for (15, 10, 2022) 14, 10, 2022
2. Calculate previous date for (1, 1, 2015) 31, 12, 2014
3. Calculate previous date for (31, 3, 2000) 30, 3, 2000

Invalid dates:

1. Calculate previous date for (29, 2, 2022) Invalid date
2. Calculate previous date for (31, 4, 2010) Invalid date
3. Calculate previous date for (30, 2, 2000) Invalid date

Out of range dates:

1. Calculate previous date for (0, 5, 2010) Invalid date
2. Calculate previous date for (15, 13, 2005) Invalid date
3. Calculate previous date for (31, 12, 1899) Invalid date

Boundary Value Analysis:

Using boundary value analysis, we can identify the following boundary test cases:

1. The earliest possible date: (1, 1, 1900)
2. The latest possible date: (31, 12, 2015)
3. The earliest day of each month: (1, 1, 2000), (1, 2, 2000), (1, 3, 2000),..., (1, 12, 2000)

4. The latest day of each month: (31, 1, 2000), (28, 2, 2000), (31, 3, 2000),..., (31, 12, 2000)
5. Leap year day: (29, 2, 2000)
6. Invalid leap year day: (29, 2, 1900)
7. One day before earliest date: (31, 12, 1899)
8. One day after latest date: (1, 1, 2016)

Based on these boundary test cases, we can design the following test cases:
Tester Action and Input Data Expected Outcome

Boundary Test Cases:

1. Calculate previous date for (1, 1, 1900) Invalid date
2. Calculate previous date for (31, 12, 2015) 30, 12, 2015
3. Calculate previous date for (1, 1, 2000) 31, 12, 1999
4. Calculate previous date for (31, 1, 2000) 30, 1, 2000
5. Calculate previous date for (29, 2, 2000) 28, 2, 2000
6. Calculate previous date for (29, 2, 1900) Invalid date
7. Calculate previous

Program 1:

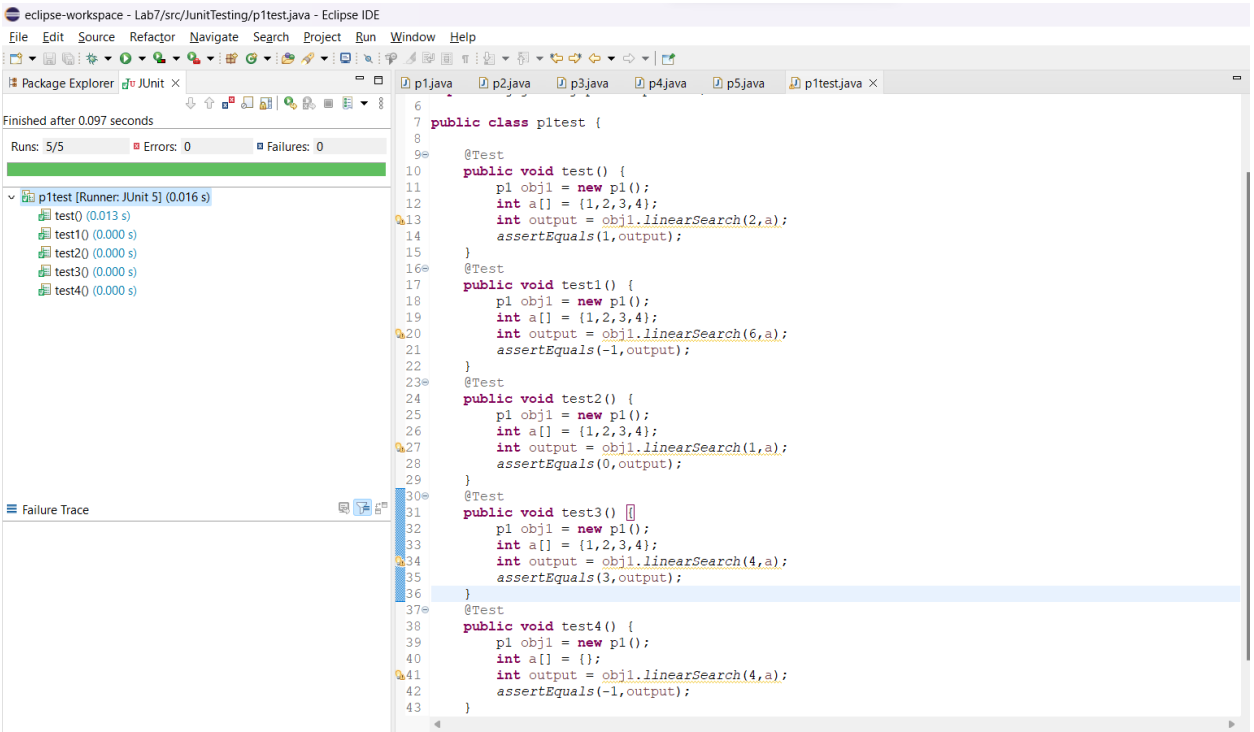
Equivalence class table

Test Case	Array	target	output	Expected

ID	value			output
1	{1,2,3,4}	2	1	1
2	{1,2,3,4}	6	-1	-1
3	{}	1	-1	-1

Boundary class table

Case id	array	target	output	Expected output
1	{1,2,3,4}	1	0	0
2	{1,2,3,4}	4	3	3



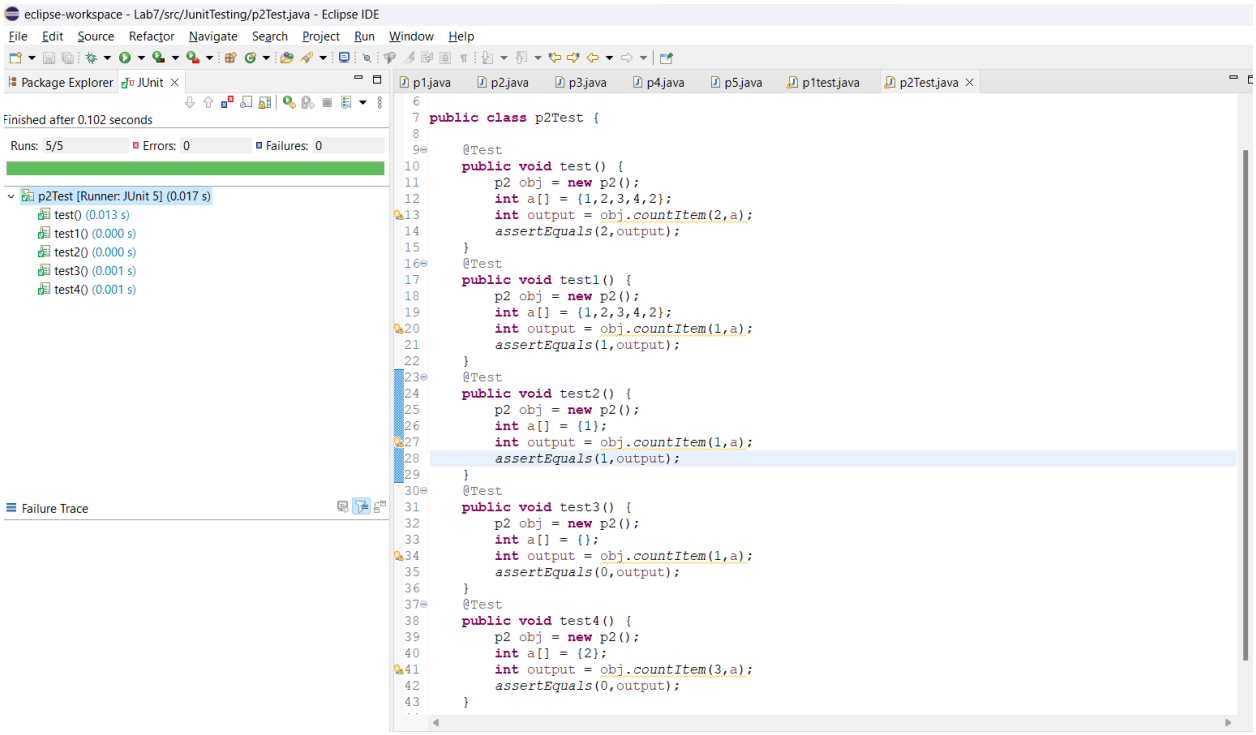
Program 2 -

Equivalence class table

Test Case ID	Array value	target	output	Expected output
1	{1,2,3,4,2}	2	2	2
2	{1,2,3,4,2}	1	1	1
3	{2}	3	0	0

Boundary class table

Case id	array	target	output	Expected output
1	{}	1	0	0
2	{1}	1	1	1



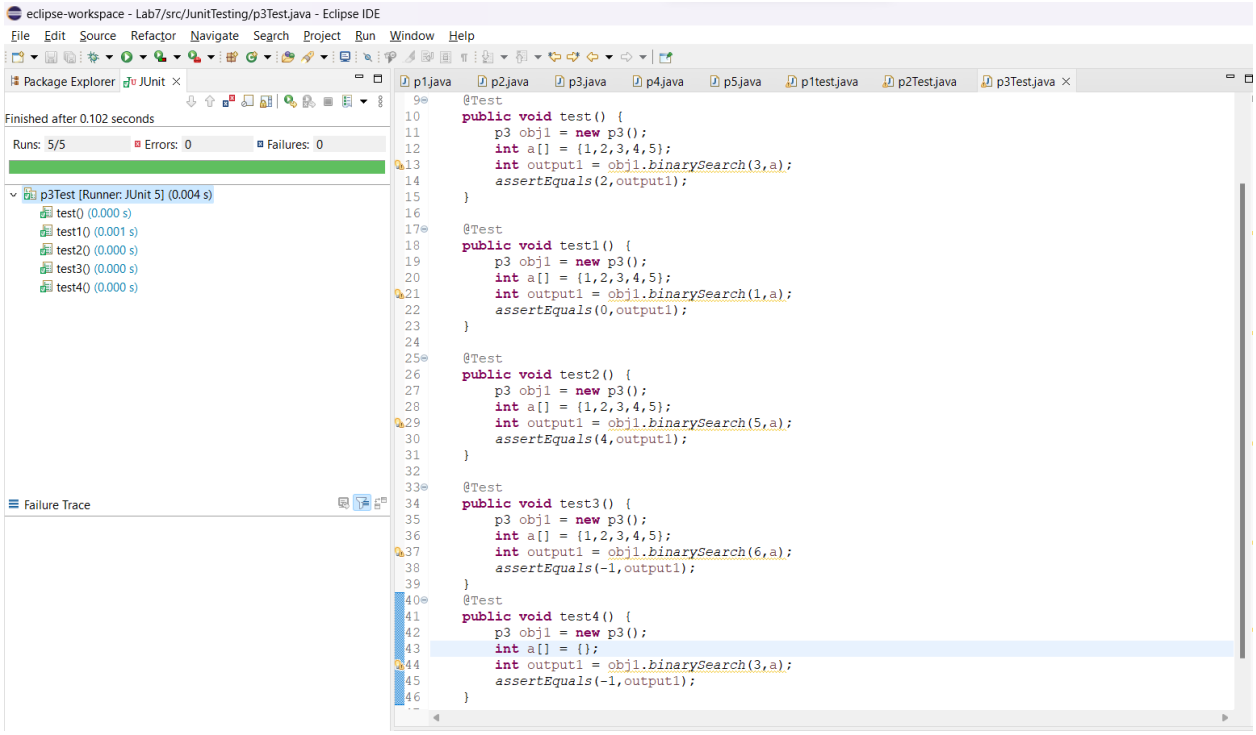
Program - 3

Equivalence class table

Test Case ID	Array value	target	output	Expected output
1	{1,2,3,4,5}	3	2	2
2	{1,2,3,4,5}	6	-1	-1
3	{}	3	-1	-1

Boundary class table

Case id	array	target	output	Expected output
1	{1,2,3,4,5}	5	4	4
2	{1,2,3,4,5}	1	0	0



Program 4 -

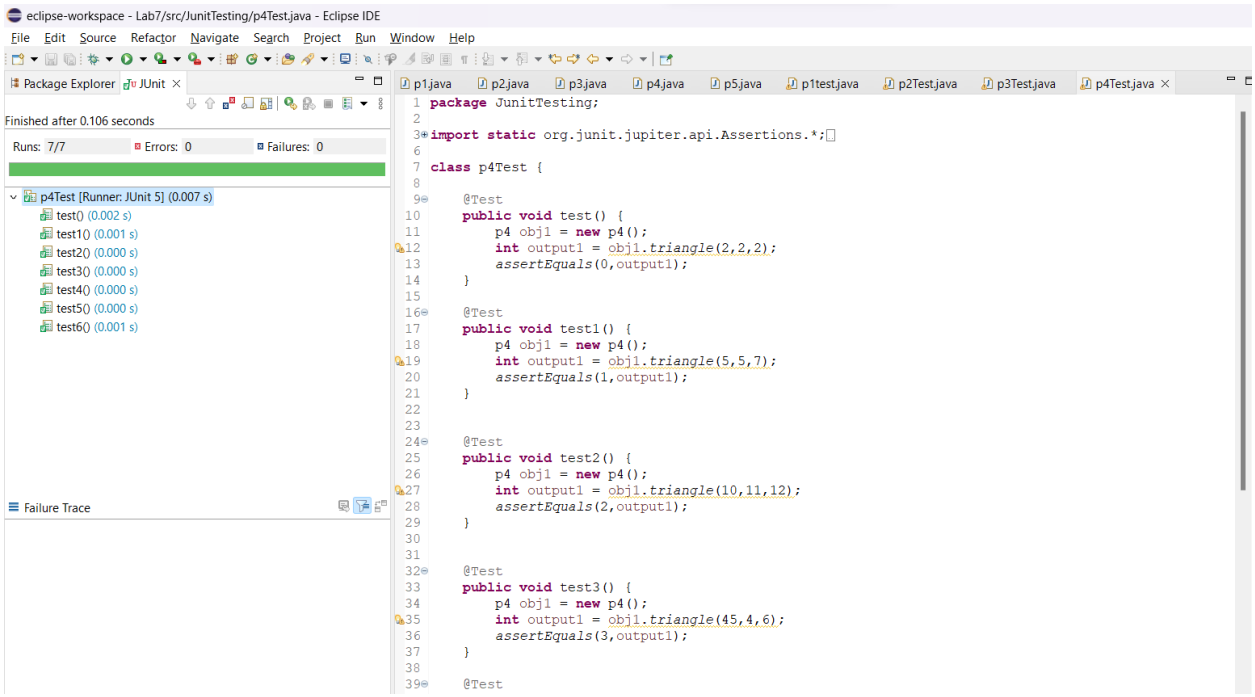
Equivalence class table

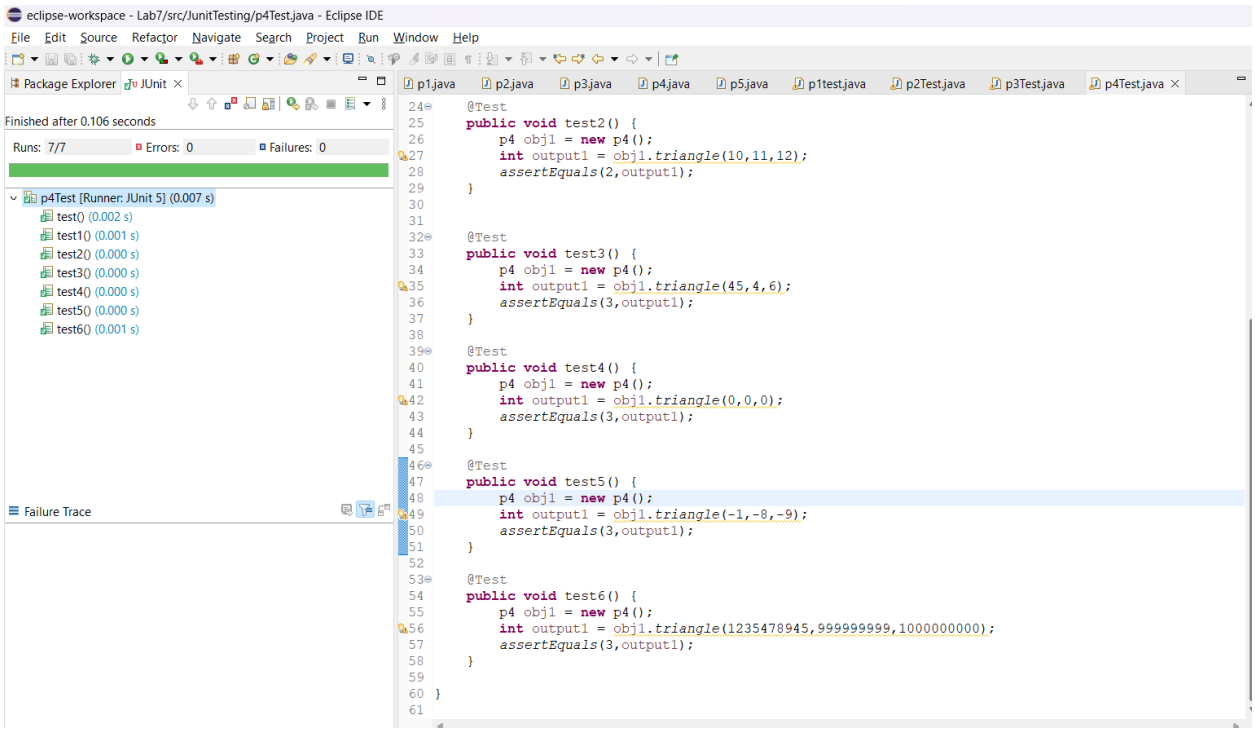
Test Case ID	Array value	output	Expected output
1	2,2,2	0	0

2	5,5,7	1	1
3	10,11,12	2	2

Boundary class table

Case id	array	output	Expected output
1	45,4,6	3	3
2	0,0,0	3	3
3	-1,-8,-9	3	3
4	1235478945,999999999,1000000000	3	3


eclipse-workspace - Lab7/src/JunitTesting/p4Test.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit
Finished after 0.106 seconds
Runs: 7/7 Errors: 0 Failures: 0
p4Test [Runner: JUnit 5] (0.007 s)
test() (0.002 s)
test1() (0.001 s)
test2() (0.000 s)
test3() (0.000 s)
test4() (0.000 s)
test5() (0.000 s)
test6() (0.001 s)
Failure Trace
p1.java p2.java p3.java p4.java p5.java p1test.java p2Test.java p3Test.java p4Test.java
package JunitTesting;
import static org.junit.jupiter.api.Assertions.*;
class p4Test {
@Test
public void test() {
p4 obj1 = new p4();
int output1 = obj1.triangle(2,2,2);
assertEquals(0,output1);
}
@Test
public void test1() {
p4 obj1 = new p4();
int output1 = obj1.triangle(5,5,7);
assertEquals(1,output1);
}
@Test
public void test2() {
p4 obj1 = new p4();
int output1 = obj1.triangle(10,11,12);
assertEquals(2,output1);
}
@Test
public void test3() {
p4 obj1 = new p4();
int output1 = obj1.triangle(45,4,6);
assertEquals(3,output1);
}
@Test


eclipse-workspace - Lab7/src/JunitTesting/p4Test.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit
Finished after 0.106 seconds
Runs: 7/7 Errors: 0 Failures: 0
p4Test [Runner: JUnit 5] (0.007 s)
test() (0.002 s)
test1() (0.001 s)
test2() (0.000 s)
test3() (0.000 s)
test4() (0.000 s)
test5() (0.000 s)
test6() (0.001 s)
Failure Trace
p1.java p2.java p3.java p4.java p5.java p1test.java p2Test.java p3Test.java p4Test.java
@Test
public void test2() {
p4 obj1 = new p4();
int output1 = obj1.triangle(10,11,12);
assertEquals(2,output1);
}
@Test
public void test3() {
p4 obj1 = new p4();
int output1 = obj1.triangle(45,4,6);
assertEquals(3,output1);
}
@Test
public void test4() {
p4 obj1 = new p4();
int output1 = obj1.triangle(0,0,0);
assertEquals(3,output1);
}
@Test
public void test5() {
p4 obj1 = new p4();
int output1 = obj1.triangle(-1,-8,-9);
assertEquals(3,output1);
}
@Test
public void test6() {
p4 obj1 = new p4();
int output1 = obj1.triangle(1235478945,999999999,1000000000);
assertEquals(3,output1);
}
}

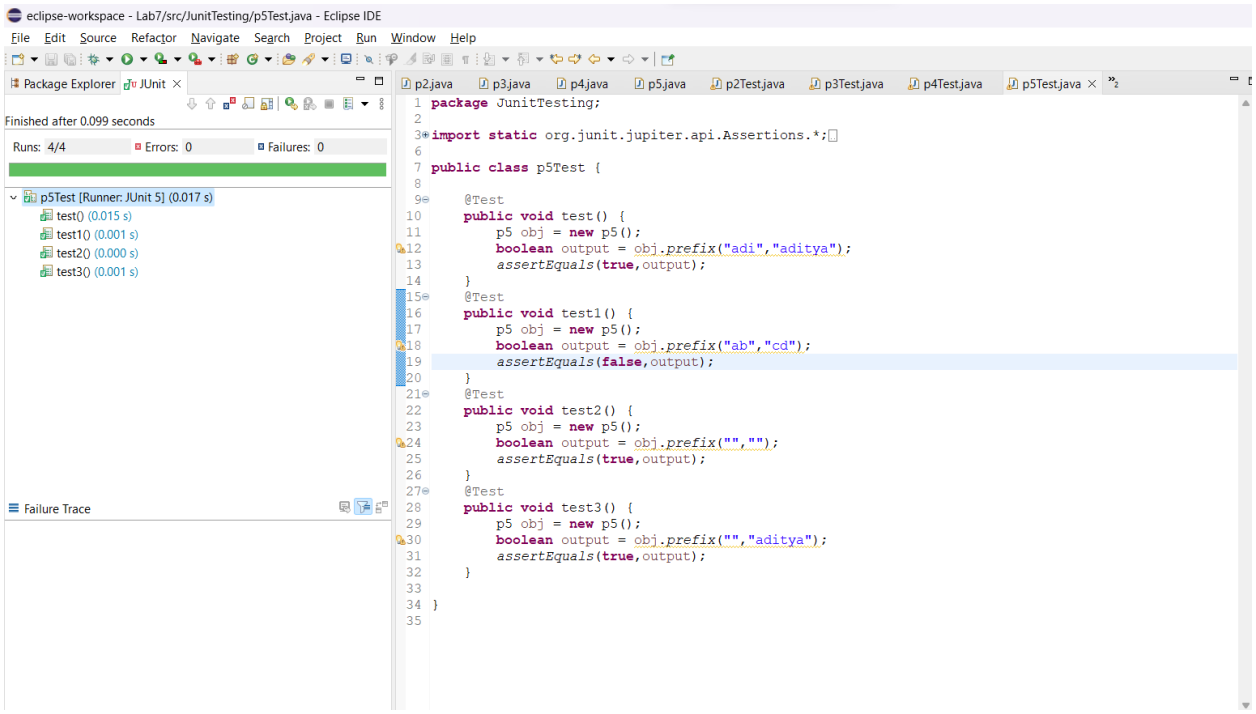
Program 5 -

Equivalence class table

Test Case ID	String Values	output	Expected output
1	“adi”,”aditya”	1	1
2	“ad”,”bc”	0	0

Boundary class table

Case id	array	target	output	Expected output
1	“”,””	1	1	1
2	“”,”aditya”	1	1	1



Program 6 -

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

EC1: All sides are real numbers and positive.
EC2: One or more than one side is/are zero or negative.
EC3: Violates the sum of side property (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.
(Hint: you must need to be ensure that the identified set of test cases cover all identified equivalence classes)

Test cases:
TC1 (EC1): A=3, B=4, C=5 (right-angled triangle)
TC2 (EC1): A=3, B=3, C=3 (equilateral triangle)
TC3 (EC1): A=10, B=11, C=12 (scalene triangle)
TC4 (EC1): A=6, B=6, C=8 (isosceles triangle)
TC5 (EC2): A=(-7), B=2, C=7 (invalid input)
TC6 (EC2): A=0, B=9, C=2 (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=1, B=3, C=5 (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=4, B=7, C=4 (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=3, B=3, C=3 (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=4, B=4, C=8 (sum of A and B is less than C)

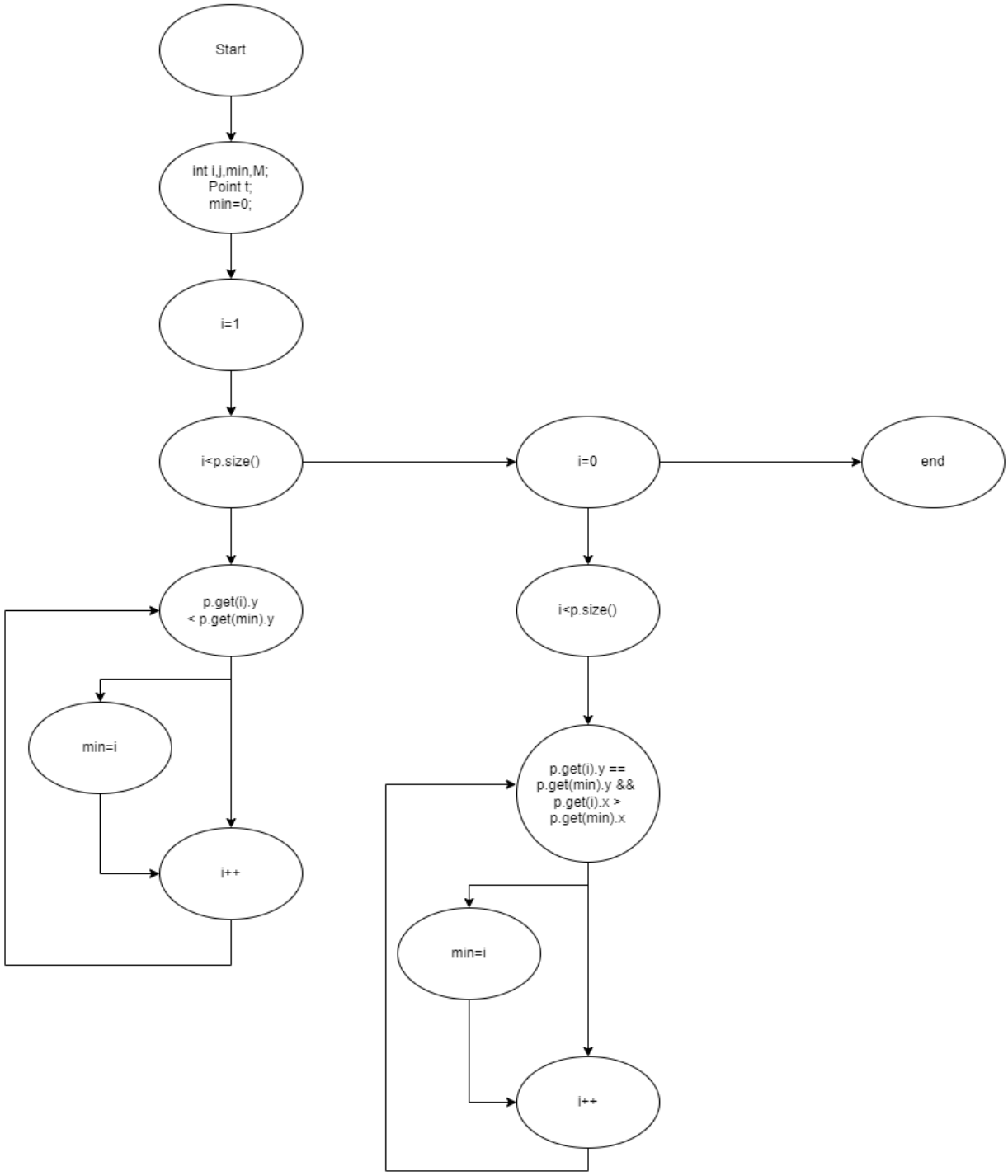
h) For non-positive input, identify test points.

Test points for non-positive input:
TP1 (EC2): A=0, B=9, C=7 (invalid input)
TP2 (EC2): A=(-2), B=9, C=4 (invalid input)

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

Section B -

Control flow diagram -



Statement coverage test cases: Every statement in code is executed at least once

Test 1: p = empty vector
Test 2: p = vector with a single point
Test 3: Two points with the same y component are in a vector, which is p.
Test 4: two points at a vector with distinct y components
Test 5: p is a vector with at least three points and distinct y components.
Test 6: p is a vector with at least three points that share the same y component.

Branch coverage test sets: Every branch in code is executed at least once
Test 1: p = empty vector
Test 2: p = vector with a single point
Test 3: Two points with the same y component are in a vector, which is p.
Test 4: two points at a vector with distinct y components
Test 5: None of the three or more points in p's vector have the same x component while all three have separate y components
Test 6: p is a vector where some of the points have the same x component and at least three of the points have the same y component.
Test 7: P is a vector with three or more points that all have the same x component and the same number of y-component points.

Basic condition coverage test sets: Every boolean expression is executed at least once

Test 1: p = empty vector
Test 2: p = vector with a single point
Test 3: The first point has a smaller x component than the second point in a vector with two points that have the same y component.
Test 4: p = vector with two points that share a y component but different x components.
Test 5: p equals a vector with two points and various y components.
Test 6: p is a vector containing at least three points with distinct y components and no two points with the same x component.
Test 7: P is a vector where at least three of the points share a common y component, and some of them also share a common x component.
Test 8: If there are three or more points in a vector that all have the same y component, then.

Examples of such test cases

Test cases :
1) p=[(x=2,y=2),(x=2,y=3),(x=1,y=3),(x=1,y=4)]
2) p=[(x=2,y=3),(x=3,y=4),(x=1,y=2),(x=5,y=6)]
3) p=[(x=1,y=5),(x=2,y=7),(x=3,y=5),(x=4,y=5),(x=5,y=6)]
4) p=[(x=1,y=2)]
5) p=[]

These 5 test cases cover all the tests discussed above.