IT314 **LAB-7** Pranav Patil
(202001055)

Section A:

1.

Tester Action and Input Data	Expected Outcome
Equivalence Partitioning	
Invalid Month (less than 1)	An Error message
Invalid Month (greater than 12)	An Error message
Valid Month, Invalid Day (less than 1)	An Error message
Valid Month, Invalid Day (greater than 31)	An Error message
Valid Month, Valid Day, Invalid Year (less than 1900)	An Error message
Valid Month, Valid Day, Invalid Year (greater than 2015)	An Error message
Valid Month, Valid Day, Valid Year	Previous date or Invalid date

Boundary Value Analysis	
Month is 1, Day is 1, Year is 1900	An Error message
Month is 12, Day is 31, Year is 2015	Previous date or Invalid date
Month is 2, Day is 29, Year is 1900	An Error message
Month is 2, Day is 28, Year is 1900	Previous date or Invalid date
Month is 2, Day is 29, Year is 2000	Previous date or Invalid date
Month is 2, Day is 28, Year is 2000	Previous date or Invalid date
Month is 4, Day is 31, Year is 2010	An Error message
Month is 4, Day is 30, Year is 2010	Previous date or Invalid date

P1.

Tester Action and Input Data	Expected Outcome
Equivalence Partitioning:	
v = 5, a = [1, 2, 3, 4, 5, 6]	4
v = 9, a = [1, 2, 3, 4, 5, 6]	-1
v = 1, a = [1, 1, 1, 1, 1, 1]	0
v = -3, a = [1, 2, 3, 4, 5, 6]	-1

Boundary Value Analysis:

v = 7, a = [1, 2, 3, 4, 5, 6] | -1
v = 1, a = [1] | 0
v = 2, a = [2] | 0
v = 5, a = [1, 2, 3, 4, 5] | 4
v = 1, a = [] | -1

Testing it in the IDE:

```java
public class LinearSearch {
    public static void main(String[] args) {
        int v = 5;
        int[] a = {1, 2, 3, 4, 5, 6};
        int result = linearSearch(v, a);
        System.out.println("Index of " + v + " in array a is " + result);

        v = 9;
        int[] b = {1, 2, 3, 4, 5, 6};
        result = linearSearch(v, b);
        System.out.println("Index of " + v + " in array b is " + result);

        v = 1;
        int[] c = {1, 1, 1, 1, 1, 1};
        result = linearSearch(v, c);
        System.out.println("Index of " + v + " in array c is " + result);

        v = -3;
        int[] d = {1, 2, 3, 4, 5, 6};
        result = linearSearch(v, d);
        System.out.println("Index of " + v + " in array d is \n " + result);
    }

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

o/p

```
java -cp /tmp/5NzHd5xaSk LinearSearch
Index of 5 in array a is 4
Index of 9 in array b is -1
Index of 1 in array c is 0
Index of -3 in array d is
 -1
```

P2.

Tester Action and Input Data	Expected Outcome
Equivalence Partitioning:	
v = 5, a = [1, 2, 3, 4, 5, 6]	1
v = 9, a = [1, 2, 3, 4, 5, 6]	0
v = 1, a = [1, 1, 1, 1, 1, 1]	6
v = -3, a = [1, 2, 3, 4, 5, 6]	0

Boundary Value Analysis:

v = 7, a = [1, 2, 3, 4, 5, 6] | 0
v = 1, a = [1] | 1
v = 2, a = [2] | 1
v = 5, a = [1, 2, 3, 4, 5] | 1
v = 1, a = [] | 0

```java
public class CountItem {
    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;
    }

    public static void main(String[] args) {
        int[] a1 = {1, 2, 3, 4, 5, 6};
        int[] a2 = {1, 2, 3, 4, 5, 6};
        int[] a3 = {1, 1, 1, 1, 1, 1};
        int[] a4 = {1, 2, 3, 4, 5, 6};
        int[] a5 = {1, 2, 3, 4, 5, 6};
        int[] a6 = {1};
        int[] a7 = {2};
        int[] a8 = {1, 2, 3, 4, 5};
        int[] a9 = {};

        System.out.println(countItem(5, a1)); // expected output: 1
        System.out.println(countItem(9, a2)); // expected output: 0
        System.out.println(countItem(1, a3)); // expected output: 6
        System.out.println(countItem(-3, a4)); // expected output: 0
        System.out.println(countItem(7, a5)); // expected output: 0
        System.out.println(countItem(1, a6)); // expected output: 1
        System.out.println(countItem(2, a7)); // expected output: 1
        System.out.println(countItem(5, a8)); // expected output: 1
        System.out.println(countItem(1, a9)); // expected output: 0
    }
}
```

o/p

```
java -cp /tmp/5NzHd5xaSk CountItem
1
0
6
0
0
1
1
1
0
```

P3.

Tester Action and Input Data	Expected Outcome
Equivalence Partitioning:	
v = 5, a = [1, 2, 3, 4, 5, 6]	4
v = 9, a = [1, 2, 3, 4, 5, 6]	-1
v = 1, a = [1, 1, 1, 1, 1, 1]	0
v = -3, a = [1, 2, 3, 4, 5, 6]	-1

Boundary Value Analysis:

v = 7, a = [1, 2, 3, 4, 5, 6] | -1
v = 1, a = [1] | 0
v = 2, a = [2] | 0
v = 5, a = [1, 2, 3, 4, 5] | 4
v = 1, a = [] | -1

```java
public class BinarySearch {
    public static void main(String[] args) {
        int[] a = {1, 2, 3, 4, 5, 6};
        int[] b = {1, 2, 3, 4, 5, 6};
        int[] c = {1, 1, 1, 1, 1, 1};
        int[] d = {1, 2, 3, 4, 5, 6};
        int[] e = {1, 2, 3, 4, 5, 6};
        int[] f = {2};
        int[] g = {1, 2, 3, 4, 5};
        int[] h = {};

        System.out.println(binarySearch(5, a));
        System.out.println(binarySearch(9, b));
        System.out.println(binarySearch(1, c));
        System.out.println(binarySearch(-3, d));
        System.out.println(binarySearch(7, e));
        System.out.println(binarySearch(1, f));
        System.out.println(binarySearch(2, g));
        System.out.println(binarySearch(5, h));
        System.out.println(binarySearch(1, h));
    }

    public static int binarySearch(int v, int[] a) {
        int lo = 0;
        int hi = a.length - 1;

        while (lo <= hi) {
            int mid = (lo + hi) / 2;

            if (v == a[mid]) {
                return mid;
            } else if (v < a[mid]) {
                hi = mid - 1;
            } else {
                lo = mid + 1;
            }
        }

        return -1;
    }
```

o/p

```
java -cp /tmp/5NzHd5xaSk BinarySearch
4
-1
2
-1
-1
-1
1
-1
-1
```

P4.

Tester Action and Input Data	Expected Outcome
Equivalence Partitioning:	
a = 2, b = 2, c = 2	EQUILATERAL
a = 5, b = 5, c = 8	ISOSCELES
a = 3, b = 4, c = 5	SCALENE
a = -1, b = -2, c = 3	INVALID
a = 0, b = 0, c = 0	INVALID
a = 2, b = 3, c = 8	INVALID

Boundary Value Analysis:

a = 1, b = 1, c = 1 | EQUILATERAL
a = 2, b = 2, c = 3 | ISOSCELES
a = 3, b = 4, c = 5 | SCALENE
a = 1, b = 2, c = 3 | INVALID
a = 100, b = 100, c = 100 | EQUILATERAL
a = 199, b = 200, c = 399 | ISOSCELES
a = 100, b = 101, c = 200 | SCALENE
a = Integer.MAX_VALUE, b = Integer.MAX_VALUE, c = Integer.MAX_VALUE | EQUILATERAL
a = Integer.MIN_VALUE, b = Integer.MIN_VALUE, c = Integer.MIN_VALUE | INVALID

```java
public class Main {
    final static int EQUILATERAL = 0;
    final static int ISOSCELES = 1;
    final static int SCALENE = 2;
    final static int INVALID = 3;

    public static void main(String[] args) {
        System.out.println(triangle(2, 2, 2)); // should print 0
        System.out.println(triangle(5, 5, 8)); // should print 1
        System.out.println(triangle(3, 4, 5)); // should print 2
        System.out.println(triangle(-1, -2, 3)); // should print 3
        System.out.println(triangle(0, 0, 0)); // should print 3
        System.out.println(triangle(2, 3, 8)); // should print 3
        System.out.println(triangle(1, 1, 1)); // should print 0
        System.out.println(triangle(2, 2, 3)); // should print 1
        System.out.println(triangle(3, 4, 5)); // should print 2
        System.out.println(triangle(1, 2, 3)); // should print 3
        System.out.println(triangle(100, 100, 100)); // should print 0
        System.out.println(triangle(199, 200, 399)); // should print 2
        System.out.println(triangle(100, 101, 200)); // should print 1
        System.out.println(triangle(Integer.MAX_VALUE, Integer.MAX_VALUE, Integer.MAX_VALUE)); // should print 0
        System.out.println(triangle(Integer.MIN_VALUE, Integer.MIN_VALUE, Integer.MIN_VALUE)); // should print 3
    }

    static 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);
    }
}
```

o/p:

```
java -cp /tmp/5NzHd5xaSk Main
0
1
2
3
3
3
0
1
2
3
0
3
2
3
0
```

P5.

Test Case	Input s1	Input s2	Expected Output
1	"hello"	"hello world"	true
2	"world"	"hello world"	false
3	"hello"	"hi"	false
4	""	"hello"	true
5	"hello"	""	false
6	"hello"	"hello"	true
7	"hello"	"helloo"	true
8	"hello"	"hell"	true

Boundary values analysis :

Test Case	Input s1	Input s2	Expected Output
1	"a"	"a"	true
2	"a"	"b"	false
3	"a"	""	false
4	""	"a"	true
5	"a"	"aa"	true
6	"a"	"aaa"	true
7	"aa"	"a"	false

```java
public class Main {
    public static void main(String[] args) {
        System.out.println(prefix("hello", "hello world")); // false
        System.out.println(prefix("world", "hello world")); // false
        System.out.println(prefix("hello", "hi")); // false
        System.out.println(prefix("", "hello")); // true
        System.out.println(prefix("hello", "")); // false
        System.out.println(prefix("hello", "hello")); // true
        System.out.println(prefix("hello", "helloo")); // true
        System.out.println(prefix("a", "a")); // true
        System.out.println(prefix("a", "b")); // false
        System.out.println(prefix("a", "")); // false
        System.out.println(prefix("", "a")); // true
    }

    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;
    }
}
```

o/p:

```
java -cp /tmp/5NzHd5xaSk Main
true
false
false
true
false
true
true
true
false
false
true
```

P6.

a) Equivalence Classes:

Equivalence Class	Condition
Valid triangle	A + B > C, B + C > A, C + A > B
Scalene triangle	All sides have different lengths
Isosceles triangle	Two sides have equal lengths
Equilateral triangle	All three sides have the same length
Right-angle triangle	One angle is equal to 90 degrees
Non-triangle	One of the conditions for a valid triangle is not met

b) Test Cases:

Test Case	Condition	Expected Outcome
1	A=5, B=5, C=5	Equilateral triangle
2	A=3, B=4, C=5	Right-angle triangle
3	A=2, B=2, C=3	Isosceles triangle
4	A=4, B=4, C=7	Scalene triangle
5	A=1, B=1, C=3	Non-triangle

c) Boundary Condition for Scalene Triangle:

Test Case	Condition	Expected Outcome
1	A=2.0, B=3.0, C=5.0	Not a valid triangle
2	A=1.0, B=2.0, C=3.0	Not a valid triangle
3	A=1.0, B=1.0, C=2.0	Not a valid triangle
4	A=0.1, B=0.1, C=0.3	Not a valid triangle
5	A=1.0, B=2.0, C=2.9	Scalene triangle (boundary value)

d) Boundary Condition for Isosceles Triangle:

Test Cases	Input Values	Expected Output
1	4 4 7	Isosceles

2	**7 4 4**	**Isosceles**
3	**4 7 4**	**Isosceles**

e) Boundary Condition for Equilateral Triangle:

Test Cases	Input Values	Expected Output
1	**1 1 1**	**Equilateral**
2	**100 100 100**	**Equilateral**
3	**0.1 0.1 0.1**	**Equilateral**

f) Boundary Condition for Right Angle Triangle:

Test Cases	Input Values	Expected Output

1	**3 4 5**	**Right Angle**
2	**5 3 4**	**Right Angle**
3	**4 5 3**	**Right Angle**
4	**6 8 10**	**Right Angle**
5	**8 10 6**	**Right Angle**
6	**10 6 8**	**Right Angle**
7	**1 1.4 1.7**	**Not Right Angle**
8	**1.4 1 1.7**	**Not Right Angle**
9	**1.7 1.4 1**	**Not Right Angle**

10	**1.414213 1 2**	**Right Angle**
11	**1 1.414213 2**	**Right Angle**
12	**1.414213 2 1**	**Right Angle**
13	**2 1.414213 1**	**Right Angle**
14	**2 1 1.414213**	**Right Angle**
15	**1 2 1.414213**	**Right Angle**
16	**1.414213 1.414213 2**	**Not Right Angle**

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

Test Case	Input Values	Expected Output

TC1	**A = 0, B = 1, C = 2**	**Not a triangle**
TC2	**A = 1, B = 0, C = 2**	**Not a triangle**
TC3	**A = 1, B = 2, C = 0**	**Not a triangle**
TC4	**A = 1, B = 2, C = 3**	**Not a triangle**
TC5	**A = 2, B = 3, C = 5**	**Not a triangle**
TC6	**A = 3, B = 4, C = 7**	**Not a triangle**

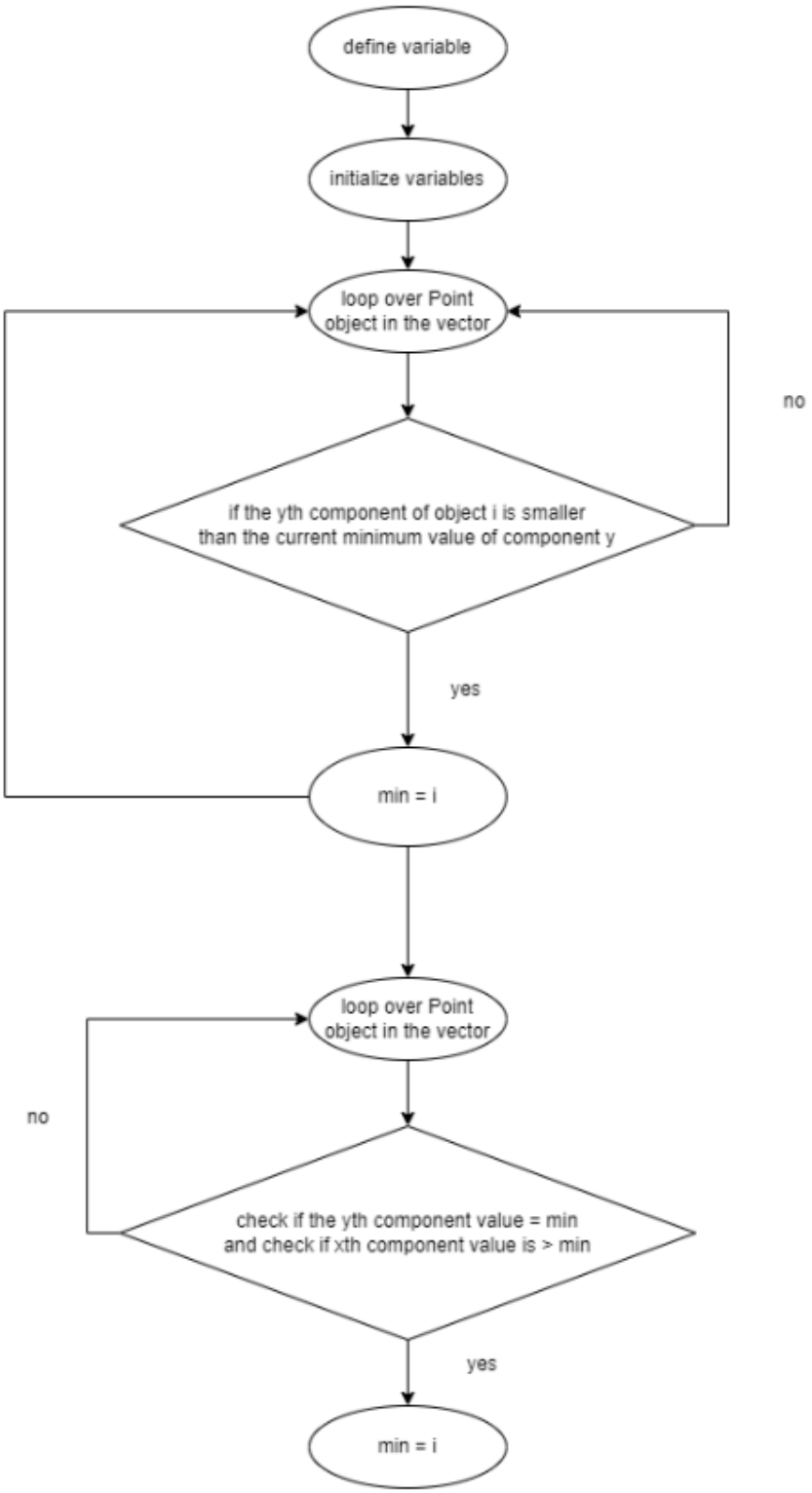
h) For non-positive input, identify test points.

Test Case	Input Values	Expected Output
TC1	**A = -1, B = 2, C = 3**	**Invalid input**
TC2	**A = 1, B = -2, C = 3**	**Invalid input**

TC3	**A = 1, B = 2, C = -3**	**Invalid input**
TC4	**A = -1, B = -2, C = -3**	**Invalid input**
TC5	**A = 0, B = 1, C = 2**	**Invalid input**

Section B:

1)


define variable
initialize variables
loop over Point object in the vector
if the yth component of object i is smaller than the current minimum value of component y
no
yes
min = i
loop over Point object in the vector
no
check if the yth component value = min and check if xth component value is > min
yes
min = i

2)

a. Statement Coverage:

Test Case	Input	Expected Output
1	p = []	Empty vector
2	p = [(1,1)]	Vector with single point
3	p = [(1,1), (2,2)]	Vector with two points
4	p = [(1,1), (2,2), (3,1)]	Vector with three points
5	p = [(1,1), (2,2), (3,1), (4,3)]	Vector with four points

b. Branch Coverage:

Test Case	Input	Expected Output
1	p = []	Empty vector

2	p = [(1,1)]	Vector with single point
3	p = [(1,1), (2,2)]	Vector with two points
4	p = [(1,1), (2,2), (3,1)]	Vector with three points
5	p = [(1,1), (2,2), (3,1), (4,3)]	Vector with four points
6	p = [(1,2), (3,1), (2,1)]	Vector with three points in different order

c. Basic Condition Coverage:

Test Case	Input	Expected Output
1	p = []	Empty vector
2	p = [(1,1)]	Vector with single point
3	p = [(1,1), (2,2)]	Vector with two points
4	p = [(1,1), (2,2), (3,1)]	Vector with three points

5	**p = [(1,1), (2,2), (3,1), (4,3)]**	**Vector with four points**
6	**p = [(1,2), (3,1), (2,1)]**	**Vector with three points in a different order**
7	**p = [(1,1), (1,1), (1,1)]**	**Vector with three identical points**
8	**p = [(1,1), (2,2), (1,1)]**	**Vector with two identical points**
9	**p = [(1,1), (1,2), (2,1)]**	**Vector with two points with same y component**