Name : Patel Naisargi | ID : 202201291

Question : 1

TestCase	**Output**
15,6,2010	14,6,2010
29,2,2012	28,2,2012
31,12,2015	30,12,2015
0,6,2010	Error
32,7,2005	Error
29,2,13	Error
15,0,2005	Error
15,13,2005	Error
15,5,2018	Error
15,5,2020	Error

Question : 1

Equivalence Partitioning :

v = 3, a = [1, 2, 3, 4]	2
v = 5, a = [1, 2, 3, 4]	-1
v = 3, a = []	-1

Boundary Value Analysis :

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

Modified Code :

```
public static int linearSearch(int v, int[] a)

{ int i = 0;

while (i < a.length)

 { if (a[i] == v)
{ return i;

}

i++;

}

return -1;

}
```

Question : 2

Equivalence Partitioning :

v = 2, a = [1, 2, 3, 2]	2
v = 2, a = [1, 2, 3]	1
v = 4, a = [1, 2, 3]	0
v = 1, a = []	0

Boundary Value Analysis :

v = 2, a = [2]	1
v = 2, a = [1]	0
v = 2, a = [2, 2, 3]	2
v = 2, a = [1, 3, 2, 2]	2

Modified Code :

```
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;
}
```

Question : 3

Equivalence Partitioning :

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

Boundary Value Analysis :

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

Modified Code :

```
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;

}
```

Question : 4

Equivalence Partitioning :

a = 3, b = 3, c = 3	EQUILATERAL
a = 5, b = 5, c = 3	ISOSCELES
a = 3, b = 4, c = 5	SCALENE
a = 1, b = 2, c = 3	INVALID
a = 0, b = 4, c = 5	INVALID
a = -1, b = 3, c = 4	INVALID

Boundary Value Analysis :

a = 1, b = 1, c = 1	EQUILATERAL
a = 10000, b = 10000, c = 10000	EQUILATERAL
a = 1, b = 2, c = 3	INVALID

Modified Code :

```
final int EQUILATERAL = 0;

final int ISOSCELES = 1;

final int SCALENE = 2;

final int INVALID = 3;

public static int triangle(int a, int b, int c) {

if (a <= 0 || b <= 0 || c <= 0 || 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;
}
```

Question : 5

Equivalence Partitioning :

s1 = "pre", s2 = "prefix"	True
s1 = "post", s2 = "prefix"	False
s1 = "longer", s2 = "short"	False
s1 = "", s2 = "anything"	True

Boundary Value Analysis :

s1 = "", s2 = ""	True
s1 = "a...a" (1000 times), s2 = "a...a" (1000 times)	True

Modified Code :

```
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;
}
```

Question : 6

(A)

Valid Triangles:

- **Equilateral Triangle**: All three sides are equal.
  - Example: A = 5.0, B = 5.0, C = 5.0
- **Isosceles Triangle**: Two sides are equal.
  - Example: A = 5.0, B = 5.0, C = 3.0
- **Scalene Triangle**: All three sides are different.
  - Example: A = 3.0, B = 4.0, C = 5.0
- **Right-angled Triangle**: Satisfies the Pythagorean theorem (A^2 + B^2 = C^2).
  - Example: A = 3.0, B = 4.0, C = 5.0 (or any permutation)

Invalid Triangles:

- **Non-Triangle (Triangle Inequality Violation)**: The sum of two sides is less than or equal to the third.
  - Example: A = 1.0, B = 2.0, C = 3.0
- **Non-Positive Sides**: One or more sides are zero or negative.
  - Example: A = 0, B = 4.0, C = 5.0
  - Example: A = -1, B = 4.0, C = 5.0

(B)

Equilateral Triangle	A = 5.0, B = 5.0, C = 5.0	All sides equal.
Isosceles Triangle	A = 5.0, B = 5.0, C = 3.0	Two sides equal.
Scalene Triangle	A = 3.0, B = 4.0, C = 5.0	All sides different, valid triangle.

Right-angled Triangle	A = 3.0, B = 4.0, C = 5.0	Satisfies Pythagorean theorem.
Non-Triangle (Inequality)	A = 1.0, B = 2.0, C = 3.0	Invalid triangle, fails inequality.
Non-positive Sides	A = 0.0, B = 4.0, C = 5.0	One side is zero.
Non-positive Sides	A = -1.0, B = 4.0, C = 5.0	One side is negative.

(C)

Boundary for A + B > C (Scalene Triangle)

- **Boundary Test 1**: A = 1.0, B = 2.0, C = 2.99999
  - Expected outcome: Valid scalene triangle (because A + B > C).
- **Boundary Test 2**: A = 1.0, B = 2.0, C = 3.0
  - Expected outcome: Invalid triangle (because A + B == C).
- **Boundary Test 3**: A = 1.0, B = 2.0, C = 3.00001
  - Expected outcome: Valid scalene triangle (because A + B > C).

(D)

Boundary for A = B (Isosceles Triangle)

- **Boundary Test 1**: A = 5.0, B = 5.0, C = 0.00001
  - Expected outcome: Isosceles triangle (two equal sides).
- **Boundary Test 2**: A = 5.0, B = 5.0, C = 5.0
  - Expected outcome: Equilateral triangle (all sides equal).
- **Boundary Test 3**: A = 5.0, B = 5.0, C = 10.0
  - Expected outcome: Invalid triangle

(E)

Boundary for A = B = C (Equilateral Triangle)

- **Boundary Test**: A = 5.0, B = 5.0, C = 5.0

    - Expected outcome: Equilateral triangle.

(F)

Boundary for $A^2 + B^2 = C^2$ (Right-angled Triangle)

- **Boundary Test 1**: A = 3.0, B = 4.0, C = 5.0
    - Expected outcome: Right-angled triangle
- **Boundary Test 2**: A = 3.0, B = 4.0, C = 5.00001
    - Expected outcome: Scalene triangle
- **Boundary Test 3**: A = 3.0, B = 4.0, C = 4.99999
    - Expected outcome: Scalene triangle

(G)

Boundary Test 1: A = 1.0, B = 1.0, C = 2.0

- Expected outcome: Invalid triangle

Boundary Test 2: A = 1.0, B = 1.0, C = 1.99999

- Expected outcome: Valid triangle

Boundary Test 3: A = 1.0, B = 1.0, C = 2.00001

- Expected outcome: Invalid triangle

(H)

Non-positive Input Test 1: A = 0.0, B = 4.0, C = 5.0

- Expected outcome: Invalid triangle.

Non-positive Input Test 2: A = -1.0, B = 4.0, C = 5.0

- Expected outcome: Invalid triangle.