Software Engineering
(IT-314)

Name - Mihir Moolchandani
ID - 202201088

Q1: Equivalence Class Test Cases for Date Validation Program

Ans:

Equivalence Partitioning:

1. **Valid Inputs:**
    - Month: 1 <= Month <= 12
    - Day: 1 <= Day <= 31 (it does varies depending on the month)
    - Year: 1900 <= Year <= 2015

    Test Cases:

Input	Expected Outcome
15,5,2010	14,5,2010
1,1,2000	31,12,1999

2. **Invalid Inputs (Equivalence Partitioning):**
    - Month > 12 or Month < 1
    - Day > 31 or Day < 1
    - Year > 2015 or Year < 1900

    Test Cases:

Input	Expected Outcome
32,5,2000	Invalid Date
31,13,2005	Invalid Date
1,0,1999	Invalid Date

3. **Edge Cases (Boundary Values):**

    Test Cases:

Input	Expected Outcome
1,1,1900	Invalid Date
1,12,2015	30,11,2015

Q.2. Programs:

Ans:

Program1: Linear Search

Equivalence Partitioning:

1. Valid Inputs: Array has the value being searched

Input	Expected Outcome
v = 5, a = [1, 2, 5, 6]	2

2. Invalid Inputs: Value not present in the array

Input	Expected Outcome
v = 7, a = [1, 2, 3, 4]	-1

Boundary Value Analysis:

1. Test Case 1: Empty array

Input	Expected Outcome
v = 3, a = [ ]	-1

2. Test Case 2: Single element array, value matches

Input	Expected Outcome
v = 4, a = [4]	0

Program 2: countItem

Equivalence Partitioning:

1. Valid Input: Value appears in the array

Input	Expected Outcome
v = 3, a = [1, 3, 3, 4, 5, 3]	3

2. Invalid Input: Value not present in the array

Input	Expected Outcome
v = 7, a = [1, 3, 6, 9]	0

Boundary Value Analysis:

1. Test Case 1: Empty array

Input	Expected Outcome
v = 3, a = [ ]	0

Program 3: `binarySearch`

Equivalence Partitioning:

1. Valid Input: Value is present in the sorted array

Input	Expected Outcome
v = 7, a = [1, 3, 5, 7, 8, 9]	3

2. Invalid Input: Value not present in the sorted array

Input	Expected Outcome
v = 2, a = [1, 3, 5, 6, 7, 8, 9]	-1

Boundary Value Analysis:

1. Test Case 1: Single element array, value present

Input	Expected Outcome
v = 4, a = [4]	0

2. Test Case 2: Empty array

Input	Expected Outcome
v = 3, a = [ ]	-1

Program 4: `triangle`

Equivalence Partitioning:

1. Valid Input: Equilateral triangle

Input	Expected Outcome
a = 3, b = 3, c = 3	EQUILATERAL

2. Invalid Input: Impossible triangle

Input	Expected Outcome
a = 1, b = 2, c = 3	INVALID

Boundary Value Analysis:

1. Test Case 1: Scalene triangle

Input	Expected Outcome
a = 3, b = 4, c = 5	SCALENE

Program 5: `prefix`

Equivalence Partitioning:

1. Valid Input: s1 is a prefix of s2

Input	Expected Outcome
s1 = “pre” s2 = “prefix”	true

2. Invalid Input: s1 is not a prefix of s2

Input	Expected Outcome
s1 = “post” s2 = “prefix”	false

Boundary Value Analysis:

1. Test Case 1: Empty string as prefix

Input	Expected Outcome
s1 = “ ” s2 = “anything”	true

Program 6: Triangle Classification (Floating-Point)

a) Identify the equivalence classes for the system

1. **Valid Triangle Classes:**
    - **Equilateral Triangle:** All three sides are equal (A = B = C).
    - **Isosceles Triangle:** Exactly two sides are equal (A = B ≠ C, A = C ≠ B, B = C ≠ A).
    - **Scalene Triangle:** All sides are different (A ≠ B ≠ C).
    - **Right-Angled Triangle:** Satisfies the Pythagorean theorem ($A^2 + B^2 = C^2$, $A^2 + C^2 = B^2$, $B^2 + C^2 = A^2$).
2. **Invalid Triangle Classes:**
    - **Non-Triangle:** The sum of the lengths of any two sides must be greater than the third side (A + B ≤ C, A + C ≤ B, B + C ≤ A).
    - **Non-Positive Input:** At least one side length is non-positive (A ≤ 0, B ≤ 0, C ≤ 0).

b) Identify test cases to cover the identified equivalence classes

1. **Equilateral Triangle:** (3, 3, 3)
2. **Isosceles Triangle:** (4, 4, 5)
3. **Scalene Triangle:** (4, 5, 6)
4. **Right-Angled Triangle:** (3, 4, 5) (Right-angled at 3, 4)
5. **Non-Triangle:** (1, 2, 3) (1 + 2 = 3)
6. **Non-Positive Input:** (0, 1, 2), (-1, 2, 3)

Input	Expected Outcome
a = 4, b = 4, c = 4	EQUILATERAL
a = 3, b = 3, c = 5	ISOSCELES
a = 3, b = 5, c = 6	SCALENE
a = 3, b = 4, c = 5	RIGHT-ANGLED
a = 3, b = 1, c = 2	INVALID
a = -1, b = 2, c = 3	INVALID

c) Boundary condition A + B > C case (scalene triangle)

1. **Test Case:** (2, 3, 4) — Valid scalene triangle (2 + 3 > 4).
2. **Test Case:** (2, 2, 4) — Invalid case (2 + 2 = 4).
3. **Test Case:** (3, 3, 5) — Valid isosceles triangle but edge of scalene (3 + 3 > 5).

Input	Expected Outcome
a = 2, b = 3, c = 4	SCALENE
a = 2, b = 2, c = 4	INVALID
a = 2, b = 2, c = 3	SCALENE

d) Boundary condition A = C case (isosceles triangle)

1. **Test Case:** (4, 5, 4) — Valid isosceles triangle.
2. **Test Case:** (4, 4, 8) — Invalid case (4 + 4 = 8).
3. **Test Case:** (5, 5, 3) — Valid isosceles triangle.

Input	Expected Outcome
a = 4, b = 5, c = 4	ISOSCELES
a = 4, b = 4, c = 8	INVALID
a = 5, b = 5, c = 3	ISOSCELES

e) Boundary condition A = B = C case (equilateral triangle)

1. **Test Case:** (5, 5, 5) — Valid equilateral triangle.
2. **Test Case:** (4, 4, 4) — Another valid equilateral triangle.
3. **Test Case:** (2, 2, 4) — Invalid case (2 + 2 = 4).

Input	Expected Outcome
a = 5, b = 5, c = 5	EQUILATERAL
a = 4, b = 4, c = 4	EQUILATERAL
a = 2, b = 2, c = 4	INVALID

f) Boundary condition $A^2 + B^2 = C^2$ case (right-angle triangle)

1. **Test Case:** (3, 4, 5) — Valid right-angled triangle.
2. **Test Case:** (6, 8, 10) — Valid right-angled triangle ($6^2 + 8^2 = 10^2$).
3. **Test Case:** (5, 12, 13) — Valid right-angled triangle.

Input	Expected Outcome
a = 3, b = 4, c = 5	RIGHT-ANGLED
a = 6, b = 8, c = 10	RIGHT-ANGLED
a = 5, b = 12, c = 13	RIGHT-ANGLED

g) Non-triangle case, identify test cases to explore the boundary

1. **Test Case:** (1, 1, 2) — Invalid case (1 + 1 = 2).
2. **Test Case:** (3, 1, 2) — Invalid case (1 + 2 < 3).
3. **Test Case:** (0, 1, 1) — Invalid due to non-positive input.

Input	Expected Outcome
a = 1, b = 1, c = 2	INVALID
a = 3, b = 1, c = 2	INVALID
a = 0, b = 1, c = 1	INVALID

h) Non-positive input, identify test points

1. **Test Case:** (0, 1, 2) — One side is zero.
2. **Test Case:** (1, -1, 2) — One side is negative.
3. **Test Case:** (-1, -1, -1) — All sides are negative.

Input	Expected Outcome
a = 0, b = 1, c = 2	INVALID
a = 1, b = -1, c = 2	INVALID
a = -1, b = -1, c = -1	INVALID