LAB-8

BLACK BOX TESTING 21/10

HERIK PATEL **202201407**

QUESTION-1:

Equivalence Class Partitioning (EP):

We will identify the valid and invalid partitions for **day**, **month**, and **year** based on the given input constraints:

Input Condition	Valid Partition	Invalid Partitions
Day	1 ≤ day ≤ 31 for valid months	- Day < 1 (e.g., 0, -5)
		- Day > 31 (e.g., 32, 45)
		- Invalid day for a specific month (e.g., 31st in April)
Month	1 ≤ month ≤ 12	- Month < 1 (e.g., 0, -2)
		- Month > 12 (e.g., 13, 15)
Year	1900 ≤ year ≤ 2015	- Year < 1900 (e.g., 1899)
		- Year > 2015 (e.g., 2020, 2025)

Test Cases Based on Equivalence Class Partitions:

Day	Month	Year	Expected Outcome	Reason
15	5	2000	Previous date: 14/5/2000	Valid input (middle of valid partitions)
0	5	2000	Error: Invalid day	Invalid day (Day < 1)
32	5	2000	Error: Invalid day	Invalid day (Day > 31)
30	2	2001	Error: Invalid day	Invalid day for February (non-leap year)
12	13	2000	Error: Invalid month	Invalid month (Month > 12)
12	0	2000	Error: Invalid month	Invalid month (Month < 1)
12	5	1899	Error: Invalid year	Invalid year (Year < 1900)
12	5	2020	Error: Invalid year	Invalid year (Year > 2015)
1	3	2000	Previous date: 29/2/2000	Valid leap year for February
31	12	2015	Previous date: 30/12/2015	Valid input (end of the upper boundary for year)

Boundary Value Analysis Test Cases:

Day:

- Lower boundary: Day = 1
- Upper boundary: Day = 31

Special boundary cases:

- February 28/29 (depends on whether it is a leap year).
- 30th day for months like April, June, September, and November.

Month:

- Lower boundary: Month = 1 (January)
- Upper boundary: Month = 12 (December)

Year:

- Lower boundary: Year = 1900 (minimum valid year)
- Upper boundary: Year = 2015 (maximum valid year)

Day	Month	Year	Expected Outcome	Boundary Condition
1	1	1900	Previous date: 31/12/1899	Lower boundary for day, month, and year
31	12	2015	Previous date: 30/12/2015	Upper boundary for day, month, and year
1	3	2000	Previous date: 29/2/2000	Leap year boundary for February
1	3	2001	Previous date: 28/2/2001	Non-leap year boundary for February
30	4	2001	Previous date: 29/4/2001	Boundary for months with 30 days
31	3	2000	Previous date: 30/3/2000	Boundary for months with 31 days
1	1	2000	Previous date: 31/12/1999	Lower boundary for year 2000
31	1	2000	Previous date: 30/1/2000	Upper boundary for January

QUESTION - 2:

Program 1: Linear Search

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

Test Case	Expected Outcome	Remarks
v = 3, a = [1, 2, 3, 4, 5]	2	Basic test case; value found at index 2.
v = 6, a = [1, 2, 3, 4, 5]	-1	Value not present in the array.
v = 1, a = [1, 2, 3, 4, 5]	0	First element matches.
v = 5, a = [1, 2, 3, 4, 5]	4	Last element matches.
v = 1, a = []	-1	Empty array; no elements to search.
v = 2, a = [1, 2, 2, 3, 4]	1	Multiple occurrences; first found at index 1.
v = -1, a = [-5, -4, -3, -2, -1]	4	Negative number search; found at index 4.
v = 0, a = [-1, -2, -3, 0, 1]	3	Searching for zero; found at index 3.

Program 2: Frequency of a value

```
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 Case	Expected Outcome	Remarks
v = 2, a = [1, 2, 3, 2, 4]	2	Value 2 appears twice in the array.
v = 5, a = [1, 2, 3, 4]	0	Value 5 is not present in the array.
v = 1, a = [1, 1, 1, 1, 1]	5	Value 1 appears five times in the array.
v = 0, a = []	0	Empty array; no elements to count.
v = -1, a = [-1, -1, 0, 1]	2	Value -1 appears twice in the array.
v = 3, a = [3, 3, 3, 3, 3]	5	Value 3 appears five times in the array.
v = 4, a = [1, 2, 3, 4, 5]	1	Value 4 appears once in the array.
v = 10, a = [10, 10, 10, 10]	4	Value 10 appears four times in the array.

Program 3: Binary Search

```
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 Case	Expected Outcome	Remarks
v = 3, a = [1, 2, 3, 4, 5]	2	Value 3 found at index 2.
v = 6, a = [1, 2, 3, 4, 5]	-1	Value 6 not present in the array.
v = 1, a = [1, 2, 3, 4, 5]	0	Value 1 found at index 0.
v = 5, a = [1, 2, 3, 4, 5]	4	Value 5 found at index 4.
v = 1, a = []	-1	Empty array; no elements to search.
v = 2, a = [1, 2, 2, 3, 4]	1	Value 2 found at index 1 (first occurrence).
v = -1, a = [-5, -4, -3, -2, -1]	4	Value -1 found at index 4.
v = 0, a = [-1, -2, -3, 0, 1]	3	Value 0 found at index 3.

Program 4: Valid Triangle

```
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 Case	Expected Outcome	Remarks
a = 3, b = 3, c = 3	0	Equilateral triangle; all sides equal.
a = 5, b = 5, c = 3	1	Isosceles triangle; two sides equal.
a = 4, b = 5, c = 6	2	Scalene triangle; all sides different.
a = 1, b = 2, c = 3	3	Invalid triangle; lengths do not form a triangle.
a = 0, b = 1, c = 1	3	Invalid triangle; side length cannot be zero.
a = 2, b = 2, c = 4	3	Invalid triangle; lengths do not satisfy triangle inequality.
a = 7, b = 7, c = 7	0	Equilateral triangle; all sides equal.
a = 3, b = 4, c = 5	2	Scalene triangle; all sides different.

Program 5: Is string 1 a prefix of string 2?

```
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 Case	Expected Outcome	Remarks
s1 = "pre", s2 = "prefix"	true	"pre" is a prefix of "prefix".
s1 = "prefix", s2 = "pre"	false	"prefix" is longer than "pre".
s1 = "abc", s2 = "abcdef"	true	"abc" is a prefix of "abcdef".
s1 = "def", s2 = "abcdef"	false	"def" is not a prefix of "abcdef".
s1 = "", s2 = "hello"	true	An empty string is a prefix of any string.
s1 = "hello", s2 = ""	false	"hello" cannot be a prefix of an empty string.
s1 = "test", s2 = "testing"	true	"test" is a prefix of "testing".
s1 = "longer", s2 = "long"	false	"longer" is longer than "long".

Program 6: Valid Triangle (Part 2)

Equivalence Classes

1. **Valid Triangles:**
   - Equilateral Triangle: All sides are equal (A = B = C).
   - Isosceles Triangle: Exactly two sides are equal (A = B, A = C, or B = C).
   - Scalene Triangle: All sides are different (A ≠ B, B ≠ C, A ≠ C).
   - Right-Angled Triangle: Follows Pythagorean theorem ($A^2 + B^2 = C^2$).
2. **Invalid Triangles:**
   - Non-Triangle: The sum of any two sides must be greater than the third side (A + B ≤ C, A + C ≤ B, or B + C ≤ A).
   - Non-positive Input: Any side length that is less than or equal to zero (A ≤ 0, B ≤ 0, C ≤ 0).

b) Test Cases to Cover Equivalence Classes

Input Values (A, B, C)	Expected Outcome	Equivalence Class
(3, 3, 3)	Equilateral	Equilateral Triangle
(5, 5, 3)	Isosceles	Isosceles Triangle
(4, 5, 6)	Scalene	Scalene Triangle
(3, 4, 5)	Right-Angled	Right-Angled Triangle
(1, 2, 3)	Invalid	Non-Triangle
(0, 1, 2)	Invalid	Non-Positive Input
(3, 3, 0)	Invalid	Non-Positive Input

c) Boundary Condition A + B > C (Scalene Triangle)

Expected Outcome	**Remarks**
Scalene	Valid triangle
Invalid	Boundary case

d) Boundary Condition A = C (Isosceles Triangle)

Input Values (A, B, C)	**Expected Outcome**	**Remarks**
(2, 3, 2)	Isosceles	Valid triangle
(0, 3, 0)	Invalid	Non-positive input

e) Boundary Condition A = B = C (Equilateral Triangle)

Input Values (A, B, C)	**Expected Outcome**	**Remarks**
(2, 2, 2)	Equilateral	Valid triangle
(0, 0, 0)	Invalid	Non-positive input

f) Boundary Condition $A^2 + B^2 = C^2$ (Right-Angled Triangle)

Input Values (A, B, C)	**Expected Outcome**	**Remarks**
(3, 4, 5)	Right-Angled	Valid triangle
(5, 12, 13)	Right-Angled	Valid triangle
(5, 5, 5)	Invalid	Not a right-angled triangle

g) Non-Triangle Case

Input Values (A, B, C)	**Expected Outcome**	**Remarks**
(1, 2, 3)	Invalid	Non-triangle
(5, 5, 15)	Invalid	Non-triangle

h) Non-Positive Input

Input Values (A, B, C)	**Expected Outcome**	**Remarks**
(0, 1, 2)	Invalid	Non-positive input
(-1, 2, 3)	Invalid	Non-positive input