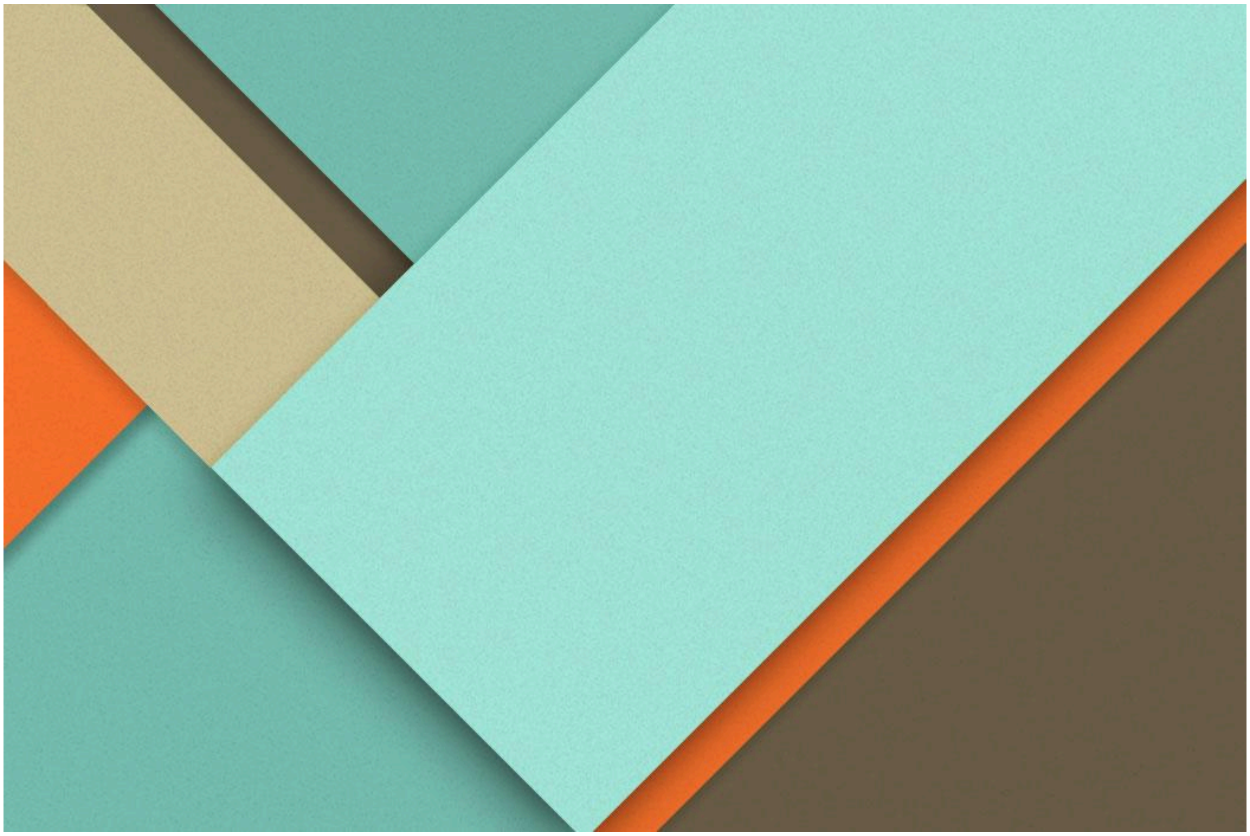


Lab08

RAMYA SHAH
202201409

Q1)

Equivalence Class Table

Equivalence Class	**Class Type**
Day < 1	Invalid
Day > 31	Invalid
1 <= Day <= 31	Valid
Month < 1	Invalid
Month > 12	Invalid
1 <= Month <= 12	Valid
Year < 1900	Invalid
Year > 2015	Invalid
1900 <= Year <= 2015	Valid

Test Cases Table based on Equivalence Partitioning

Test Case ID	**Input (Day, Month, Year)**	**Equivalence Class**	**Expected Output**
TC1	(0, 1, 2000)	Day < 1	Error message
TC2	(32, 1, 2000)	Day > 31	Error message
TC3	(15, 5, 2000)	1 <= Day <= 31	14/5/2000
TC4	(15, 0, 2000)	Month < 1	Error message
TC5	(15, 13, 2000)	Month > 12	Error message
TC6	(15, 5, 2000)	1 <= Month <= 12	14/5/2000
TC7	(15, 5, 1899)	Year < 1900	Error message

Test Case ID	Input (Day, Month, Year)	Equivalence Class	Expected Output
TC8	(15, 5, 2016)	Year > 2015	Error message
TC9	(15, 5, 2000)	1900 <= Year <= 2015	14/5/2000

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:

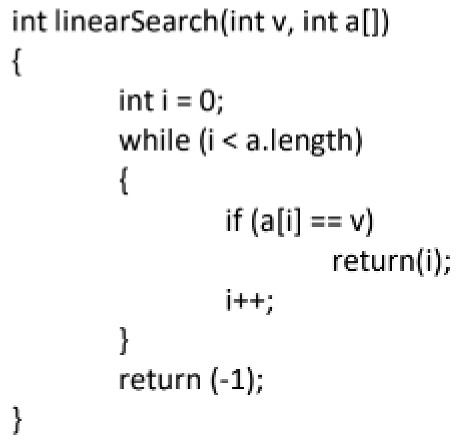
- Lower boundary: Year = 1900 (minimum valid year)
- Upper boundary: Year = 2015 (maximum valid year)

Boundary Value Analysis (BVA) Test Cases

Test Case ID	**Input (Day, Month, Year)**	**Boundary Condition**	**Expected Output**
TC1	(1, 1, 1900)	Lower boundary: Day = 1, Month = 1, Year = 1900	31/12/1899
TC2	(31, 12, 2015)	Upper boundary: Day = 31, Month = 12, Year = 2015	30/12/2015
TC3	(1, 3, 2000)	Special case: Day = 1, Month = 2 (Leap year, Feb 29)	29/2/2000
TC4	(28, 2, 2001)	Special boundary: February 28 (non-leap year)	27/2/2001
TC5	(29, 2, 2004)	Special boundary: February 29 (leap year)	28/2/2004
TC6	(30, 4, 2000)	Special boundary: 30th day (April)	29/4/2000
TC7	(30, 6, 2000)	Special boundary: 30th day (June)	29/6/2000
TC8	(30, 9, 2000)	Special boundary: 30th day (September)	29/9/2000
TC9	(30, 11, 2000)	Special boundary: 30th day (November)	29/11/2000
TC10	(1, 1, 2015)	Lower boundary: Day = 1, Year = 2015	31/12/2014

Q2) Program

Program 1: Linear Search



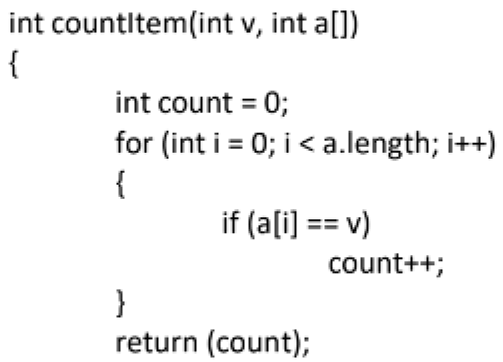
```
int linearSearch(int v, int a[])
{
        int i = 0;
        while (i < a.length)
        {
                if (a[i] == v)
                        return(i);
                i++;
        }
        return (-1);
}
```

Test Cases Table for `linearSearch`

Test Case ID	Input (v, a)	Description	Expected Output
TC1	(5, [1, 2, 3, 4, 5])	Value exists at the last position	4
TC2	(3, [1, 2, 3, 4, 5])	Value exists in the middle	2
TC3	(1, [1, 2, 3, 4, 5])	Value exists at the first position	0
TC4	(6, [1, 2, 3, 4, 5])	Value does not exist in the array	-1

Test Case ID	**Input (v, a)**	**Description**	**Expected Output**
TC5	(3, [3])	Array has one element, value exists	0
TC6	(2, [3])	Array has one element, value does not exist	-1
TC7	(2, [])	Array is empty	-1
TC8	(-1, [-1, 0, 1, 2])	Value is negative and exists in the array	0
TC9	(0, [-3, -2, 0, 3, 4])	Value is zero and exists in the array	2
TC10	(7, [7, 7, 7, 7])	Array contains repeated elements	0

Program 2: Frequency of a value



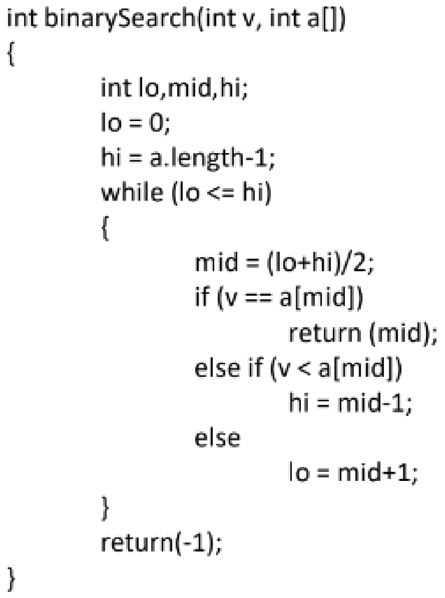
```
int countItem(int v, int a[])
{
        int count = 0;
        for (int i = 0; i < a.length; i++)
        {
                if (a[i] == v)
                        count++;
        }
        return (count);
```

Test Cases Table for `countItem`

Test Case ID	**Input (v, a)**	**Description**	**Expected Output**
TC1	(2, [1, 2, 2, 3, 2])	Value appears multiple times	3
TC2	(3, [1, 2, 3, 4, 5])	Value appears once	1
TC3	(1, [1, 2, 3, 4, 5])	Value appears once at the start	1
TC4	(5, [1, 2, 3, 4, 5])	Value appears once at the end	1
TC5	(6, [1, 2, 3, 4, 5])	Value does not exist in the array	0

Test Case ID	Input (v, a)	Description	Expected Output
TC6	(1, [1, 1, 1, 1, 1])	All elements match the value	5
TC7	(0, [0, 0, 0, 0])	Value is zero, all elements match	4
TC8	(-1, [-1, 0, 1, -1])	Value is negative and appears twice	2
TC9	(2, [])	Array is empty	0
TC10	(7, [1, 2, 3, 4, 5])	Value is not present	0

Program 3: Binary Search



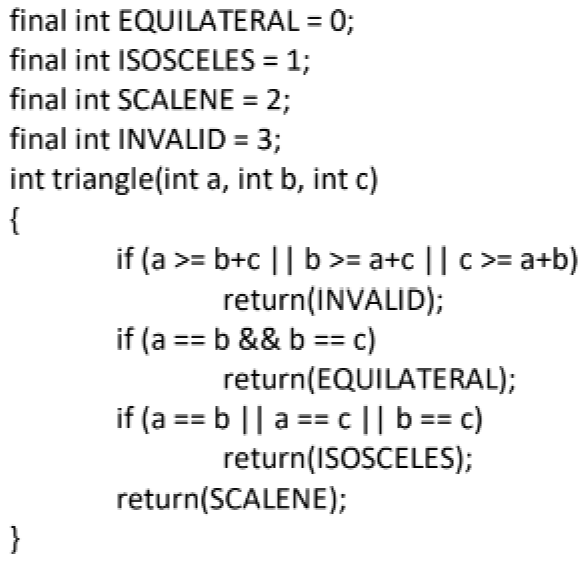
```
int binarySearch(int v, int a[])
{
        int lo,mid,hi;
        lo = 0;
        hi = a.length-1;
        while (lo <= hi)
        {
                mid = (lo+hi)/2;
                if (v == a[mid])
                        return (mid);
                else if (v < a[mid])
                        hi = mid-1;
                else
                        lo = mid+1;
        }
        return(-1);
}
```

Test Cases Table for `binarySearch`

Test Case ID	Input (v, a)	Description	Expected Output
TC1	(3, [1, 2, 3, 4, 5])	Value exists in the array	2
TC2	(1, [1, 2, 3, 4, 5])	Value is at the first position	0
TC3	(5, [1, 2, 3, 4, 5])	Value is at the last position	4
TC4	(6, [1, 2, 3, 4, 5])	Value does not exist in the array	-1
TC5	(0, [1, 2, 3, 4, 5])	Value is less than all elements	-1
TC6	(4, [1, 2, 3, 4, 5])	Value is in the middle	3
TC7	(3, [1, 1, 3, 3, 5])	Value appears multiple times	2
TC8	(7, [1, 2, 3, 4, 5, 6])	Value is greater than all elements	-1

Test Case ID	**Input (v, a)**	**Description**	**Expected Output**
TC9	(2, [2, 3, 4, 5, 6])	Value is at the first position	0
TC10	(3, [])	Array is empty	-1

Program 4: Valid Triangle



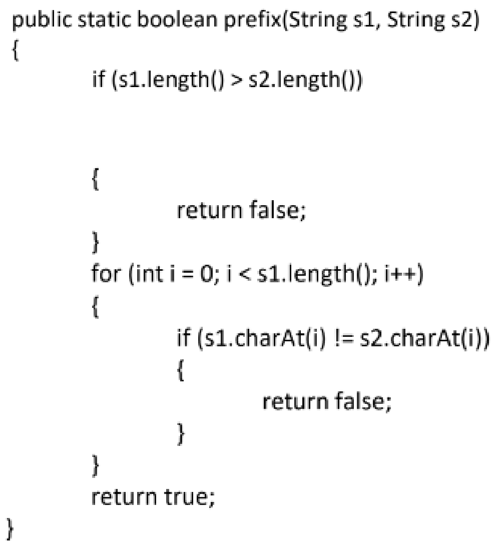
```
final int EQUILATERAL = 0;
final int ISOSCELES = 1;
final int SCALENE = 2;
final int INVALID = 3;
int triangle(int a, int b, int c)
{
        if (a >= b+c || b >= a+c || c >= a+b)
                return(INVALID);
        if (a == b && b == c)
                return(EQUILATERAL);
        if (a == b || a == c || b == c)
                return(ISOSCELES);
        return(SCALENE);
}
```

Test Cases Table for `triangle`

Test Case ID	**Input (a, b, c)**	**Description**	**Expected Output**
TC1	(3, 3, 3)	Equilateral triangle	0
TC2	(5, 5, 3)	Isosceles triangle (two sides equal)	1
TC3	(4, 5, 6)	Scalene triangle (no sides equal)	2
TC4	(1, 2, 3)	Invalid triangle (does not satisfy triangle inequality)	3
TC5	(0, 0, 0)	Invalid triangle (all sides are zero)	3
TC6	(2, 2, 4)	Invalid triangle (two sides do not add up to more than the third)	3
TC7	(7, 10, 5)	Scalene triangle	2

Test Case ID	**Input (a, b, c)**	**Description**	**Expected Output**
TC8	(2, 2, 2)	Equilateral triangle	0
TC9	(2, 2, 3)	Isosceles triangle	1
TC10	(10, 1, 1)	Invalid triangle (two sides are too short)	3

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 Cases Table for `prefix`

Test Case ID	Input (s1, s2)	Description	Expected Output
TC1	("pre", "prefix")	`s1` is a valid prefix of `s2`	true
TC2	("pre", "presentation")	`s1` is a valid prefix of `s2`	true
TC3	("prefix", "prefix")	`s1` is exactly the same as `s2`	true
TC4	("test", "testing")	`s1` is a valid prefix of `s2`	true
TC5	("test", "best")	`s1` is not a prefix of `s2`	false
TC6	("abc", "abcd")	`s1` is a valid prefix of `s2`	true
TC7	("abc", "ab")	`s1` is longer than `s2`, so it cannot be a prefix	false

Test Case ID	Input (s1, s2)	Description	Expected Output
TC8	("", "non-empty")	Empty string s1 is a prefix of any non-empty string	true
TC9	("non-empty", "")	s1 is longer than s2, cannot be a prefix	false
TC10	("ABC", "abc")	Case-sensitive check, s1 is not a prefix	false

Program 6: Valid Triangle (Part 2)

a) Equivalence Classes for Triangle Classification

Equivalence Class	**Description**
Valid Equilateral Triangle	All sides are equal ($A = B = C$)
Valid Isosceles Triangle	Exactly two sides are equal ($A = B$ or $A = C$ or $B = C$)
Valid Scalene Triangle	All sides are different ($A \neq B, B \neq C, A \neq C$)
Valid Right-Angled Triangle	Follows Pythagorean theorem ($A^2 + B^2 = C^2$)
Invalid Triangle	Non-Triangle: $A + B \leq C$ or $A + C \leq B$ or $B + C \leq A$
Non-Positive Input	Any side length is less than or equal to zero ($A \leq 0.0, B \leq 0.0, C \leq 0.0$)

b)Test Cases Covering Identified Equivalence Classes

Test Case ID	**Input (A, B, C)**	**Equivalence Class**	**Expected Output**
TC1	(5.5, 5.5, 5.5)	Valid Equilateral Triangle	"Equilateral"
TC2	(5.5, 5.5, 4.4)	Valid Isosceles Triangle	"Isosceles"
TC3	(4.5, 5.5, 6.7)	Valid Scalene Triangle	"Scalene"
TC4	(3.0, 4.0, 5.0)	Valid Right-Angled Triangle	"Right-Angled"
TC5	(1.0, 2.0, 3.0)	Invalid Triangle (Non-Triangle)	"Invalid"
TC6	(0.0, 4.4, 5.5)	Non-Positive Input	"Invalid"
TC7	(5.5, 0.0, 7.2)	Non-Positive Input	"Invalid"

Test Case ID	Input (A, B, C)	Equivalence Class	Expected Output
TC8	(8.0, 15.5, 17.0)	Valid Right-Angled Triangle	"Right-Angled"
TC9	(-3.0, 4.4, 5.5)	Non-Positive Input	"Invalid"
TC10	(10.5, 5.5, 5.5)	Valid Isosceles Triangle	"Isosceles"

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

Test Case ID	Input (A, B, C)	Description	Expected Output
TC1	(1.5, 1.5, 2.5)	All sides equal, becomes equilateral	"Equilateral"
TC2	(2.2, 2.2, 3.0)	Exactly two sides equal, becomes isosceles	"Isosceles"
TC3	(2.5, 3.5, 4.5)	Valid scalene triangle, A + B > C	"Scalene"
TC4	(5.5, 4.4, 10.0)	Invalid triangle (A + B ≤ C)	"Invalid"

d)Boundary Condition: A = C (Isosceles Triangle)

Test Case ID	Input (A, B, C)	Description	Expected Output
TC1	(2.5, 3.0, 2.5)	Two sides are equal, valid isosceles triangle	"Isosceles"
TC2	(5.5, 5.5, 5.5)	All sides equal, becomes equilateral	"Equilateral"
TC3	(4.4, 2.2, 4.4)	Two sides equal, valid isosceles triangle	"Isosceles"
TC4	(10.5, 5.5, 10.5)	Valid isosceles triangle	"Isosceles"

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

Test Case ID	Input (A, B, C)	Description	Expected Output
TC1	(3.3, 3.3, 3.3)	All sides equal, valid equilateral triangle	"Equilateral"
TC2	(5.5, 5.5, 5.5)	All sides equal, valid equilateral triangle	"Equilateral"
TC3	(0.0, 0.0, 0.0)	Invalid triangle (non-positive input)	"Invalid"

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

Test Case ID	Input (A, B, C)	Description	Expected Output
TC1	(3.0, 4.0, 5.0)	Valid right-angled triangle ($3.0^2 + 4.0^2 = 5.0^2$)	"Right-Angled"
TC2	(5.0, 12.0, 13.0)	Valid right-angled triangle ($5.0^2 + 12.0^2 = 13.0^2$)	"Right-Angled"
TC3	(1.5, 1.5, 2.1)	Invalid triangle (not a right-angled triangle)	"Invalid"
TC4	(8.0, 15.0, 17.0)	Valid right-angled triangle ($8.0^2 + 15.0^2 = 17.0^2$)	"Right-Angled"

g) Non-Triangle Case (Exploring Boundary)

Test Case ID	Input (A, B, C)	Description	Expected Output
TC1	(1.5, 2.5, 4.0)	Invalid triangle ($1.5 + 2.5 \leq 4.0$)	"Invalid"
TC2	(3.0, 5.0, 9.0)	Invalid triangle ($3.0 + 5.0 \leq 9.0$)	"Invalid"
TC3	(0.0, 0.0, 1.0)	Invalid triangle (non-positive input)	"Invalid"

Test Case ID	Input (A, B, C)	Description	Expected Output
TC4	(10.5, 5.5, 5.5)	Valid isosceles triangle, check boundary conditions	"Isosceles"

h) Non-Positive Input Test Points

Test Case ID	Input (A, B, C)	Description	Expected Output
TC1	(0.0, 5.5, 5.5)	Invalid triangle (A ≤ 0.0)	"Invalid"
TC2	(3.0, 0.0, 4.5)	Invalid triangle (B ≤ 0.0)	"Invalid"
TC3	(-1.5, 5.5, 5.5)	Invalid triangle (A ≤ 0.0)	"Invalid"