IT314 SOFTWARE ENGINEERING

LAB – 8 REPORT

LATHIGARA PRIYANSH – 202201449

Q1: Designing Equivalence Class Test Cases for a "Previous Date" Program

Given that the program inputs day, month, and year with the following ranges:

- 1 <= day <= 31
- 1 <= month <= 12
- 1900 <= year <= 2015

We need to design test cases for valid and invalid inputs using Equivalence Class Partitioning (ECP) and Boundary Value Analysis (BVA).

Equivalence Class Partitioning (ECP)

ECP divides input data into valid and invalid ranges based on conditions. For this problem, ECP involves identifying valid and invalid inputs for day, month, and year.

ECP Conditions:

1. Valid Day (1 <= day <= 31)
- Valid ECP: day is in the range [1, 31]
- Invalid ECP: day < 1 or day > 31 (e.g., 0, 32)

2. Valid Month (1 <= month <= 12)
- Valid ECP: month is in the range [1, 12]
- Invalid ECP: month < 1 or month > 12 (e.g., 0, 13)

3. Valid Year (1900 <= year <= 2015)
- Valid ECP: year is in the range [1900, 2015]
- Invalid ECP: year < 1900 or year > 2015 (e.g., 1899, 2016)

Boundary Value Analysis (BVA)

BVA tests the values at the boundaries of input ranges, which are often the points where errors occur.

BVA Conditions:

1. Day Boundary:
- Minimum Valid Boundary: day = 1
- Maximum Valid Boundary: day = 31
- Invalid Boundary: day = 0, day = 32

2. Month Boundary:
- Minimum Valid Boundary: month = 1
- Maximum Valid Boundary: month = 12
- Invalid Boundary: month = 0, month = 13

3. Year Boundary:
• Minimum Valid Boundary: year = 1900
• Maximum Valid Boundary: year = 2015
• Invalid Boundary: year = 1899, year = 2016

Test Cases for ECP and BVA

Testcase #	Input (Day, Month, Year)	Expected Outcome	Type (ECP or BVA)
TC1	(15, 6, 2005)	June 14, 2005	ECP (valid)
TC2	(32, 12, 2015)	Invalid Date	ECP (invalid)
TC3	(0, 7, 1999)	Invalid Date	ECP (invalid)
TC4	(15, 0, 2005)	Invalid Date	ECP (invalid)
TC5	(15, 13, 2010)	Invalid Date	ECP (invalid)
TC6	(15, 6, 1899)	Invalid Date	ECP (invalid)
TC7	(15, 6, 2016)	Invalid Date	ECP (invalid)
TC8	(1, 1, 1900)	December 31, 1899	BVA
TC9	(31, 12, 2015)	December 30, 2015	BVA
TC10	(15, 1, 2010)	January 14, 2010	BVA
TC11	(15, 12, 2010)	December 14, 2010	BVA
TC12	(15, 6, 1900)	June 14, 1900	BVA
TC13	(15, 6, 2015)	June 14, 2015	BVA

Q2: Test the Provided Programs

This section covers test cases for the provided programs (linearSearch, countItem, binarySearch, triangle, and prefix) using both valid and invalid inputs. At least five test cases are designed for each program.

P1: linearSearch Function

This function returns the first index of the value 'v' in an array 'a[]'. If the value is found, it returns the index; otherwise, it returns -1.

Testcase #	Input (v, a[])	Expected Output	Type
TC1	v = 5, a = [1, 2, 3, 4, 5]	4 (index of 5)	Valid
TC2	v = 7, a = [1, 2, 3, 4, 5]	-1 (not present)	Valid
TC3	v = 1, a = [1, 2, 3, 4, 5]	0 (index of 1)	Valid
TC4	v = 4, a = [1, 2, 3]	-1 (not present)	Invalid
TC5	v = 0, a = []	-1 (empty array)	Invalid

P2: countItem Function

This function returns the count of the value 'v' in an array 'a[]'.

Testcase #	Input (v, a[])	Expected Output	Type
TC1	v = 5, a = [5, 5, 5, 5]	4	Valid
TC2	v = 3, a = [1, 2, 3, 4, 3, 3]	3	Valid
TC3	v = 1, a = [1, 2, 1, 1, 2]	3	Valid
TC4	v = 2, a = [3, 4, 5]	0 (not present)	Valid
TC5	v = 4, a = []	0 (empty array)	Invalid

P3: binarySearch Function

This function performs binary search for a value 'v' in a sorted array 'a[]'. If the value is found, it returns the index; otherwise, it returns -1.

Testcase #	Input (v, a[])	Expected Output	Type
TC1	v = 5, a = [1, 2, 3, 4, 5]	4 (index of 5)	Valid
TC2	v = 1, a = [1, 2, 3, 4, 5]	0 (index of 1)	Valid
TC3	v = 6, a = [1, 2, 3, 4, 5]	-1 (not present)	Valid
TC4	v = 0, a = []	-1 (empty array)	Invalid
TC5	v = 3, a = [3, 3, 3, 3]	0 (first match)	Valid

P4: triangle Function

This function classifies a triangle based on the side lengths 'a', 'b', and 'c'. The output could be equilateral, isosceles, scalene, or invalid.

Testcase #	Input (a, b, c)	Expected Output	Type
TC1	a = 3, b = 3, c = 3	EQUILATERAL	Valid
TC2	a = 3, b = 4, c = 5	SCALENE	Valid
TC3	a = 1, b = 2, c = 3	INVALID	Invalid
TC4	a = 5, b = 5, c = 3	ISOSCELES	Valid
TC5	a = 0, b = 5, c = 5	INVALID (non-positive)	Invalid

P5: prefix Function

This function checks whether string 's1' is a prefix of string 's2'. If 's1' is a prefix, it returns true; otherwise, it returns false.

Testcase #	Input (s1, s2)	Expected Output	Type
TC1	s1 = "pre", s2 = "prefix"	true	Valid
TC2	s1 = "fix", s2 = "prefix"	false	Valid
TC3	s1 = "", s2 = "anything"	true (empty prefix)	Valid
TC4	s1 = "prefix", s2 = "pre"	false	Invalid
TC5	s1 = "test", s2 = ""	false (empty target)	Invalid

P6: Extended Triangle Classification (Floating Point)

This program reads floating-point values for the side lengths 'A', 'B', and 'C' of a triangle. It classifies the triangle as scalene, isosceles, equilateral, or right-angled, or returns invalid if the sides do not form a valid triangle.

Equivalence classes and boundary conditions for the triangle classification problem (with floating-point values) are identified as follows:

- Valid triangle (scalene, isosceles, equilateral): A + B > C
- Invalid triangle: A + B <= C
- Right-angle triangle: $A^2 + B^2 = C^2$ (or its permutations)
- Non-triangle: When the sum of two sides is less than or equal to the third side
- Non-positive sides: Any side length <= 0

Testcase #	Input (A, B, C)	Expected Output	Type
TC1	A = 5.0, B = 5.0, C = 5.0	EQUILATERAL	Valid
TC2	A = 3.0, B = 4.0, C = 5.0	SCALENE and RIGHT-ANGLE	Valid

TC3	A = 3.0, B = 3.0, C = 5.0	ISOSCELES	Valid
TC4	A = 1.0, B = 1.0, C = 2.5	INVALID (non-triangle)	Invalid
TC5	A = -1.0, B = 2.0, C = 2.0	INVALID (non-positive side)	Invalid