IT314 LAB-08

DETROJA AVI-202201452

Q.1

Previous Date Program: Equivalence Class and Boundary Value Test Cases

1. Equivalence Partitioning (EP)

Equivalence Partitioning divides inputs into valid and invalid sets (or "partitions"). The test cases check values within valid ranges and outside them.

Input Ranges:

Day: 1 <= day <= 31

Month: 1 <= month <= 12

Year: 1900 <= year <= 2015

Equivalence Classes Identified:

Valid Date:

Month: 1 <= month <= 12

Day: 1 <= day <= 28 (for February), 30 or 31 (for other months)

Year: 1900 <= year <= 2015

Invalid Date:

Day: < 1 or > 31

Month: < 1 or > 12

Year: < 1900 or > 2015

Equivalence Partitioning Test Cases:

Test Case	Day	Month	Year	Expected Outcome
TC1	15	5	2000	14-05-2000
TC2	1	1	1900	Invalid Date
TC3	32	12	2010	Invalid Date
TC4	31	2	2012	Invalid Date
TC5	0	5	1999	Invalid Date
TC6	30	13	1999	Invalid Date
TC7	31	5	2015	30-05-2015

2. Boundary Value Analysis (BVA)

Boundary Value Analysis tests at the edges of input ranges (boundaries). It focuses on values at or around the edges of valid ranges.

Boundary Conditions:

Day: 1 and 31 (where applicable)

Month: 1 and 12

Year: 1900 and 2015

Leap Year Consideration: February has 29 days in a leap year.

Boundary Value Test Cases:

Test Case	Day	Month	Year	Expected Outcome
TC1	1	1	1900	Invalid Date
TC2	31	12	2015	30-12-2015
TC3	29	2	2000	28-02-2000
TC4	28	2	1999	27-02-1999
TC5	1	3	2001	28-02-2001
TC6	31	5	2015	30-05-2015

Equivalence Partitioning: Focuses on valid and invalid ranges within the domain.

Boundary Value Analysis: Tests input values at the boundaries (edges) of valid ranges.

Q.2

P1. The function linearSearch searches for a value v in an array of integers a. If v appears in the array a, then the function returns the first index i, such that a[i] == v; otherwise, -1 is returned.

1. Equivalence Partitioning (EP)

Equivalence Partitioning divides input into valid and invalid partitions. In the case of linear search, the array a[] and the search value v form the input. The partitions can be:

Valid Partition:

Value v is present in the array.

Value v is not present in the array.

Invalid Partition:

Empty array.

Single-element array where v is either present or absent.

Arrays of different lengths.

Equivalence Partitioning Test Cases:

Test Case	Array a[]	Value v	Expected Outcome
TC1	[1, 2, 3, 4, 5]	3	2
TC2	[1, 2, 3, 4, 5]	6	-1
TC3	[]	3	-1
TC4	[10]	10	0
TC5	[10]	5	-1
TC6	[7, 8, 9, 10, 11, 12]	10	3

2. Boundary Value Analysis (BVA)

Boundary Value Analysis focuses on the boundaries of input values. The boundary conditions for linearSearch are related to the array length and the positions of the element v.

Boundary Conditions:

Array Length:

Minimum length (empty array, single-element array)

Maximum length (array with several elements)

Position of v:

First position

Last position

Middle position

Boundary Value Test Cases:

Test Case	Array a[]	Value v	Expected Outcome
TC7	[5, 10, 15, 20]	5	0
TC8	[5, 10, 15, 20]	20	3
TC9	[5, 10, 15, 20]	15	2
TC10	[100]	100	0
TC11	[5, 10, 15, 20, 25]	5	0

Equivalence Partitioning focuses on testing the search function with:

Arrays where the search value is present or absent.

Edge cases like an empty array or a single-element array.

Boundary Value Analysis ensures that the search is tested at the boundaries of:

Array lengths.

Positions of the search value (v), such as the first, middle, and last positions.

P2. The function countItem returns the number of times a value v appears in an array of integers a.

1. Equivalence Partitioning (EP)

Equivalence Partitioning divides input into valid and invalid partitions. In the case of countItem, the partitions will depend on the number of times the value v appears in the array a[] and the array characteristics.

Equivalence Classes Identified:

Valid Partition:

Value v appears one or more times in the array.

Value v does not appear in the array.

Invalid Partition:

Empty array.

Single-element array where v is either present or absent.

Arrays of different lengths.

Equivalence Partitioning Test Cases:

Test Case	Array a[]	Value v	Expected Outcome
TC1	[1, 2, 3, 4, 5]	3	1
TC2	[1, 2, 3, 4, 5]	6	0
TC3	[]	3	0
TC4	[10]	10	1
TC5	[10]	5	0
TC6	[7, 7, 7, 7, 7]	7	5
TC7	[3, 3, 3, 2, 2, 2, 1]	3	3

2. Boundary Value Analysis (BVA)

Boundary Value Analysis focuses on the boundaries of input values, such as the size of the array and the frequency of v.

Boundary Conditions:

Array Length:

Empty array (length = 0)

Single-element array

Large arrays with varying counts of v

Occurrences of v:

v occurs 0 times.

v occurs 1 time.

v occurs maximum possible times (all elements are v).

Boundary Value Test Cases:

Test Case	Array a[]	Value v	Expected Outcome
TC8	[5]	5	1
TC9	[10]	5	0
TC10	[5, 5, 5, 5, 5, 5, 5]	5	7
TC11	[1, 2, 3, 4, 5, 6, 7]	8	0

Equivalence Partitioning tests both valid and invalid partitions, ensuring that the function can handle different array sizes and whether v appears or not.

Boundary Value Analysis tests critical boundaries like the minimum and maximum occurrences of v, as well as edge cases like an empty or single-element array.

P3. The function binarySearch searches for a value v in an ordered array of integers a. If v appears in the array a, then the function returns an index i, such that a[i] == v; otherwise, -1 is returned.

1. Equivalence Partitioning (EP)

Equivalence Partitioning divides inputs into valid and invalid partitions. For binarySearch, the partitions depend on the presence of the value v in the array a[] and the characteristics of the array.

Equivalence Classes Identified:

Valid Partition:

Value v is present in the array.

Value v is not present in the array.

Invalid Partition:

Empty array.

Single-element array where v is either present or absent.

Arrays of varying lengths.

Equivalence Partitioning Test Cases:

Test Case	Array a[]	Value v	Expected Outcome
TC1	[1, 2, 3, 4, 5]	3	2
TC2	[1, 2, 3, 4, 5]	6	-1
TC3	[]	3	-1
TC4	[10]	10	0
TC5	[10]	5	-1
TC6	[2, 4, 6, 8, 10]	8	3

2. Boundary Value Analysis (BVA)

Boundary Value Analysis focuses on the boundaries of input values, such as the size of the array and the position of v.

Boundary Conditions:

Array Length:

Minimum length (empty array, single-element array).

Large array.

Position of v:

v at the first position.

v at the last position.

v in the middle.

Sorted Input Array:

Ensure the array is sorted, as binary search only works on ordered arrays.

Boundary Value Test Cases:

Test Case	Array a[]	Value v	Expected Outcome
TC7	[5, 10, 15, 20]	5	0
TC8	[5, 10, 15, 20]	20	3
TC9	[5, 10, 15, 20]	15	2
TC10	[100]	100	0
TC11	[1,2,3,4,5,6,7,8,9]	4	3

Equivalence Partitioning tests both valid and invalid partitions, ensuring that the function can handle arrays of different sizes and whether the value v is present or not.

Boundary Value Analysis tests critical boundaries like the position of v (first, middle, last) and edge cases such as an empty or single-element array.

P4. The following problem has been adapted from The Art of Software Testing, by G. Myers (1979). The function triangle takes three integer parameters that are interpreted as the lengths of the sides of a triangle. It returns whether the triangle is equilateral (three lengths equal), isosceles (two lengths equal), scalene (no lengths equal), or invalid (impossible lengths).

1. Equivalence Partitioning (EP)

Equivalence Partitioning divides inputs into valid and invalid sets (or "partitions"). The test cases check values from these partitions, ensuring the function behaves as expected within valid and invalid ranges.

Input Ranges:

a, b, c (triangle sides): All sides must be positive integers, and the sum of any two sides must be greater than the third side.

Equivalence Classes Identified:

Valid Triangles:
- Equilateral: a == b == c
- Isosceles: a == b || a == c || b == c

- Scalene: a ≠ b ≠ c

Invalid Triangles:
- Any set where the sum of two sides is less than or equal to the third.

Equivalence Partitioning Test Cases:

Test Case	Side 'a'	Side 'b'	Side 'c'	Expected Outcome
TC1	5	5	5	Equilateral
TC2	6	6	10	Isosceles
TC3	3	4	5	Scalene
TC4	10	1	1	Invalid
TC5	0	3	4	Invalid
TC6	-1	5	5	Invalid
TC7	8	8	20	Invalid
TC8	1	2	3	Invalid

2. Boundary Value Analysis (BVA)

Boundary Value Analysis tests values at the boundaries of input ranges. This helps ensure the function handles edge cases correctly.

Boundary Conditions:

Minimum and maximum values for sides: The smallest triangle has sides of 1, 1, 1. There are no strict upper bounds, but typical cases might use values up to some large integer.

Special condition: The sum of two sides must always be greater than the third side.

Boundary Value Test Cases:

Test Case	Side 'a'	Side 'b'	Side 'c'	Expected Outcome
TC1	1	1	1	Equilateral
TC2	1	1	2	Invalid
TC3	1000	1000	1999	Invalid
TC4	999	999	999	Equilateral
TC5	1	2	3	Invalid
TC6	2	2	3	Isosceles
TC7	1	1000	1000	Isosceles
TC8	1	2	4	Invalid

Equivalence Partitioning: Focuses on separating input into valid and invalid partitions (equilateral, isosceles, scalene, and invalid triangles).

Boundary Value Analysis: Tests extreme or edge values, such as the smallest and largest possible sides, or values that are just inside and outside valid ranges.

P5. The function prefix (String s1, String s2) returns whether or not the string s1 is a prefix of string s2(you may assume that neither s1 nor s2 is null).

1. Equivalence Partitioning (EP)

Equivalence Partitioning divides inputs into valid and invalid sets (or "partitions"). The test cases check whether the input strings behave correctly under various conditions.

Input Conditions:

Valid Inputs: s1 is a prefix of s2.
Invalid Inputs:
- s1 is not a prefix of s2.
- s1 is longer than s2.

Equivalence Partitioning Test Cases:

Test Case	s1	s2	Expected Outcome
TC1	“abc”	“abcdef”	True
TC2	“hello”	“hello world”	True
TC3	“abc"	“ab"	False
TC4	“test”	“testing”	True
TC5	“abc”	“defabc”	False
TC6	“world”	“hello world”	False
TC7	“”	“anything"	True
TC8	“a very long”	“a very long sentence”	True

2. Boundary Value Analysis (BVA)

Boundary Value Analysis tests values at the boundaries of input ranges to ensure the function handles edge cases correctly.

Boundary Conditions:

Empty String Condition: One or both strings can be empty.
Length Condition: Check cases where s1 and s2 are at their minimum and maximum lengths.

Boundary Value Test Cases:

Test Case	s1	s2	Expected Outcome
TC1	“”	“”	True
TC2	“”	“test”	True
TC3	“test”	“”	False
TC4	“a"	“a”	True
TC5	“a”	“b”	False
TC6	“prefix"	“prefix”	True
TC7	“longstring”	“long"	False
TC8	“p”	“prefix”	True

Equivalence Partitioning: Focuses on separating input into valid and invalid sets based on whether s1 is a prefix of s2.

Boundary Value Analysis: Tests extreme or edge values like empty strings, equal strings, or cases where s1 is longer than s2.

P6: Consider again the triangle classification program (P4) with a slightly different specification: The program reads floating values from the standard input. The three values A, B, and C are interpreted as representing the lengths of the sides of a triangle. The program then prints a message to the standard output that states whether the triangle, if it can be formed, is scalene, isosceles, equilateral, or right angled. Determine the following for the above program:

a) Identify Equivalence Classes

Valid Equivalence Classes:
1. Scalene Triangle: All three sides are unequal, and they satisfy the triangle inequality.
2. Isosceles Triangle: Two sides are equal, and the triangle inequality is satisfied.
3. Equilateral Triangle: All three sides are equal, and the triangle inequality is satisfied.
4. Right-Angled Triangle: The triangle satisfies the Pythagorean theorem ($A^2 + B^2 = C^2$).

Invalid Equivalence Classes:
1. Non-Triangle (A + B <= C): The sum of two sides is less than or equal to the third.
2. Non-Positive Input: One or more sides are zero or negative.

b) Test Cases to Cover Identified Equivalence Classes

Test Case	A	B	C	Expected Outcome / Equivalence Class
TC1	3.0	4.0	5.0	Right-angled triangle (valid)

TC2	5.0	5.0	5.0	Equilateral triangle (valid)
TC3	5.0	5.0	7.0	Isosceles triangle (valid)
TC4	3.0	4.0	6.5	Scalene triangle (valid)
TC5	1.0	2.0	3.5	Invalid triangle (invalid)
TC6	3.0	4.0	8.0	Invalid triangle (invalid)
TC7	-3.0	4.0	5.0	Invalid input (invalid)
TC8	0.0	4.0	5.0	Invalid input (invalid)

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

Test Case	A	B	C	Expected Outcome / Explanation
TC9	3.0	3.0	6.0	Invalid triangle (A + B = C)
TC10	3.0	3.0	5.9	Scalene triangle (A + B > C just valid)

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

Test Case	A	B	C	Expected Outcome / Explanation

TC11	5.0	3.0	5.0	Isosceles triangle with A = C
TC12	5.0	5.0	3.0	Isosceles triangle with A = B

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

Test Case	A	B	C	Expected Outcome / Explanation
TC13	5.0	5.0	5.0	Equilateral triangle (All three sides equal)

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

Test Case	A	B	C	Expected Outcome / Explanation
TC14	3.0	4.0	5.0	Right-angled triangle (classic)
TC15	5.0	12.0	13.0	Right-angled triangle (Pythagorean triple)

g) Boundary for Non-Triangle Case (A + B <= C)

Test Case	A	B	C	Expected Outcome / Explanation
TC16	1.0	2.0	3.0	Invalid triangle (A + B = C)
TC17	2.0	3.0	6.0	Invalid triangle (A + B < C)

h) Non-Positive Input

Test Case	A	B	C	Expected Outcome / Explanation
TC18	0.0	4.0	5.0	Invalid input (One side is zero)
TC19	-3.0	4.0	5.0	Invalid input (One side is negative)
TC20	-1.0	-1.0	-1.0	Invalid input (All sides are negative)