Software Engineering -IT314

Functional Testing

Joshi Sneh
202201048

Question : 1

1. Equivalence Partition:

Equivalence Classes

- **Valid Input Classes:**
  - Day: 1 ≤ day ≤ 31
  - Month: 1 ≤ month ≤ 12
  - Year: 1900 ≤ year ≤ 2500
- **Invalid Input Classes:**
  - Day: day<1 or day>31
  - Month: month < 1 or month > 12
  - Year: year<1900 or year>2015

2. Boundary Value Analysis

Boundary Values:

- **Day boundaries:** 1, 31
- **Month boundaries:** 1, 12
- **Year boundaries:** 1900, 2015

Test Case No.	Day	Month	Year	Expected Output	Reason
1	2	1	1900	(1, 1, 1900)	Valid (Normal case: previous day in same month)
2	1	3	2015	(28, 2, 2015)	Valid (Leap year February handling)
3	1	5	1990	(30, 4, 1990)	Valid (Previous month case)

4	1	1	2015	(31, 12, 2014)	Valid (Year change case)
5	29	2	2012	(28, 2, 2012)	Valid (Leap year case)
6	0	2	2000	Invalid Date	Invalid (Day < 1)
7	32	1	2010	Invalid Date	Invalid (Day > 31)
8	31	4	2001	Invalid Date	Invalid (April has 30 days)
9	29	2	2013	Invalid Date	Invalid (Non-leap year February)
10	15	13	2010	Invalid Date	Invalid (Month > 12)
11	10	5	1899	Invalid Date	Invalid (Year < 1900)
12	5	6	2016	Invalid Date	Invalid (Year > 2015)

Question 2:

P1.

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

1. Equivalence Class Partitioning

Valid Equivalence Classes:

- **Class 1**: v is present in the array a.
- **Class 2**: v is not present in the array a.
- **Class 3**: The array a contains one or more elements.

Invalid Equivalence Classes:

- **Class 4**: The array a is empty.

2. Boundary Value Analysis

- **Boundary Case 1**: Empty array (a with size 0).
- **Boundary Case 2**: Array with one element.
- **Boundary Case 3**: Array with two elements (minimum non-trivial size).
- **Boundary Case 4**: Array with a large number of elements.
- **Boundary Case 5**: v is at the first index (index 0).
- **Boundary Case 6**: v is at the last index (index a.length - 1).

Test Case No.	Tester Action and Input Data	Expected Outcome	Category Type
1	Input: 3, [1, 2, 3, 4, 5]	2 (value found at index 2)	Valid Equivalence Class 1
2	Input: 7, [1, 2, 3, 4, 5]	-1 (value not found)	Valid Equivalence Class 2
3	Input: 1, [1, 1, 1, 1]	0 (first occurrence at index 0)	Valid Equivalence Class 1
4	Input: 10, []	-1 (empty array)	Invalid Equivalence Class 4
5	Input: 5, [5]	0 (found at index 0)	Boundary Case 2
6	Input: 3, [1, 2]	-1 (value not found)	Boundary Case 3
7	Input: 1, [1, 2]	0 (value found at index 0)	Boundary Case 3
8	Input: 1, [1, 2, 3, ..., 1000]	0 (found at first index)	Boundary Case 5
9	Input: 1000, [1, 2, ..., 1000]	999 (found at last index)	Boundary Case 6
10	Input: 0, [1, 2, 3]	-1 (value not found)	Boundary Case 1
11	Input: 2, [1, 2, 2]	1 (first occurrence at index 1)	Valid Equivalence Class 1

.

P2.

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

Equivalence Partitioning (EP):

Valid Equivalence Classes:

- **Class EP1:** v is in the array a[] multiple times.
- **Class EP2:** v is in the array a[] exactly once.
- **Class EP3:** v is not in the array a[].
- **Class EP4:** a[] is empty (no elements).

Invalid Equivalence Classes:

- **Class EP5:** v is not an integer.
- **Class EP6:** a[] contains non-integer elements (invalid input for an array of integers).

Boundary Value Analysis (BVA):

Boundary Conditions for a[]:

- **Class BVA1:** a[] has a single element.
- **Class BVA2:** a[] has multiple elements.
- **Class BVA3:** a[] has no elements (length = 0).

Boundary Conditions for v:

- **Class BVA4:** v is the minimum possible integer value.
- **Class BVA5:** v is the maximum possible integer value.

Equivalence Partitioning Test Cases:

Tester Action and Input Data	Expected Outcome	Equivalence Class
v = 5, a = [1, 5, 3, 5, 7]	2	EP1
v = 3, a = [1, 2, 3, 4, 5]	1	EP2
v = 9, a = [1, 2, 3, 4, 5]	0	EP3
v = 2, a = []	0	EP4
v = 'two', a = [1, 2, 3]	Error	EP5
v = 2, a = [1, 2, 'three']	Error	EP6

Boundary Value Analysis Test Cases:

Tester Action and Input Data	Expected Outcome	Equivalence Class
v = 5, a = [5]	1	BVA1
v = 4, a = [4, 4, 4, 4, 4]	5	BVA2
v = 1, a = []	0	BVA3
v = Integer.MIN_VALUE, a = [-1, -2, -3]	0	BVA4
v = Integer.MAX_VALUE, a = [1, 2, 3]	0	BVA5

P3.

```
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);
}
```

Equivalence Partitioning (EP):

For binarySearch, we can define equivalence classes based on the input array a[] and the search value v:

1. **Valid Equivalence Classes:**
    - **Class EP1:** v exists in the array a[].
    - **Class EP2:** v does not exist in the array a[].
    - **Class EP3:** a[] is empty (no elements).
2. **Invalid Equivalence Classes:**
    - **Class EP4:** v is not an integer.
    - **Class EP5:** a[] contains non-integer elements (invalid input for an array of integers

Boundary Value Analysis (BVA):

1. **Boundary Conditions for a[]:**
    - **Class BVA1:** a[] has a single element.

- **Class BVA2:** a[] has multiple elements.
- **Class BVA3:** a[] has no elements (length = 0).

2. **Boundary Conditions for v:**
   - **Class BVA4:** v is the minimum value in the array.
   - **Class BVA5:** v is the maximum value in the array.
   - **Class BVA6:** v is smaller than the minimum value in the array.
   - **Class BVA7:** v is larger than the maximum value in the array.

Equivalence Partitioning Test Cases:

Tester Action and Input Data	Expected Outcome	Equivalence Class
v = 4, a = [1, 2, 3, 4, 5]	3 (index of 4)	EP1
v = 6, a = [1, 2, 3, 4, 5]	-1	EP2
v = 3, a = []	-1	EP3
v = 'three', a = [1, 2, 3]	Error	EP4
v = 2, a = [1, 'two', 3]	Error	EP5

Boundary Value Analysis Test Cases:

Tester Action and Input Data	Expected Outcome	Equivalence Class
v = 5, a = [5]	0 (index of 5)	BVA1
v = 3, a = [1, 2, 3, 4, 5]	2	BVA2
v = 2, a = []	-1	BVA3
v = 1, a = [1, 2, 3, 4, 5]	0	BVA4
v = 5, a = [1, 2, 3, 4, 5]	4	BVA5
v = 0, a = [1, 2, 3, 4, 5]	-1	BVA6
v = 6, a = [1, 2, 3, 4, 5]	-1	BVA7

P4.

```
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);
}
```

Equivalence Partitioning (EP)

For the triangle function, the equivalence classes are based on the different types of triangles and invalid inputs.

1. **Valid Equivalence Classes:**
    - **Class EP1:** All sides are equal (equilateral triangle).
    - **Class EP2:** Two sides are equal (isosceles triangle).
    - **Class EP3:** No sides are equal (scalene triangle).
2. **Invalid Equivalence Classes:**
    - **Class EP4:** One or more sides have a length less than or equal to zero.
    - **Class EP5:** The triangle inequality theorem is violated (one side is greater than or equal to the sum of the other two sides).

Boundary Value Analysis (BVA)

For boundary value analysis, we focus on testing values near the boundaries of valid triangles, particularly around the triangle inequality and side length limits.

1. **Boundary Conditions for Side Lengths:**
    - **Class BVA1:** The side lengths are at the minimum valid values (positive non-zero).
    - **Class BVA2:** One side is slightly less than or equal to the sum of the other two sides (violating the triangle inequality).
    - **Class BVA3:** All sides are equal.
    - **Class BVA4:** Two sides are equal, and the third is different.

Equivalence Partitioning Test Cases:

Tester Action and Input Data	Expected Outcome	Equivalence Class
a = 3, b = 3, c = 3	EQUILATERAL (0)	EP1
a = 5, b = 5, c = 8	ISOSCELES (1)	EP2
a = 3, b = 4, c = 5	SCALENE (2)	EP3
a = 0, b = 5, c = 7	INVALID (3)	EP4
a = 10, b = 5, c = 4	INVALID (3)	EP5

Boundary Value Analysis Test Cases:

Tester Action and Input Data	Expected Outcome	Equivalence Class
a = 1, b = 1, c = 1	EQUILATERAL (0)	BVA1
a = 2, b = 1, c = 1	INVALID (3)	BVA2
a = 5, b = 5, c = 5	EQUILATERAL (0)	BVA3
a = 6, b = 6, c = 7	ISOSCELES (1)	BVA4

P5.

```
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;
}
```

Equivalence Partitioning (EP)

For the prefix function, we can define the equivalence classes based on the relationship between the strings s1 and s2.

1. **Valid Equivalence Classes:**
    - **Class EP1:** s1 is a prefix of s2.
    - **Class EP2:** s1 is equal to s2.
    - **Class EP3:** s1 is not a prefix of s2 (different characters).
    - **Class EP4:** s1 is an empty string and s2 is not empty.

2. **Invalid Equivalence Classes:**
    - **Class EP5:** s1 is longer than s2 (impossible for s1 to be a prefix).
    - **Class EP6:** Both s1 and s2 are empty strings (though not null, this case is typically defined).

Boundary Value Analysis (BVA)

For boundary value analysis, we consider the lengths of the strings and edge cases involving prefixes.

1. **Boundary Conditions for String Lengths:**
    - **Class BVA1:** s1 is an empty string and s2 is not.
    - **Class BVA2:** s1 has a length of 1, and s2 has a length of 1.
    - **Class BVA3:** s1 has a length of 1, and s2 has a length greater than 1.
    - **Class BVA4:** s1 is equal to s2 (both have the same length).

Equivalence Partitioning Test Cases:

Tester Action and Input Data	Expected Outcome	Equivalence Class
s1 = "pre", s2 = "prefix"	True	EP1
s1 = "abc", s2 = "abc"	True	EP2
s1 = "prefix", s2 = "pre"	False	EP3
s1 = "", s2 = "test"	True	EP4
s1 = "test", s2 = "t"	False	EP5
s1 = "", s2 = ""	true	EP6

Boundary Value Analysis Test Cases:

Tester Action and Input Data	Expected Outcome	Equivalence Class
s1 = "", s2 = "non-empty"	True	BVA1
s1 = "a", s2 = "a"	True	BVA2
s1 = "a", s2 = "abc"	True	BVA3
s1 = "test", s2 = "test"	True	BVA4

P6.

a) Identify the Equivalence Classes

1. **Valid Equivalence Classes:**
    - **Class EP1:** Equilateral triangle (all three sides are equal).
    - **Class EP2:** Isosceles triangle (two sides are equal).
    - **Class EP3:** Scalene triangle (all sides are different).
    - **Class EP4:** Right-angled triangle (satisfies Pythagorean theorem).
2. **Invalid Equivalence Classes:**
    - **Class EP5:** Non-triangle (the lengths do not satisfy the triangle inequality).
    - **Class EP6:** Non-positive input (one or more sides are zero or negative).

b) Identify Test Cases to Cover the Identified Equivalence Classes

Tester Action and Input Data	Expected Outcome	Equivalence Class
A = 5.0, B = 5.0, C = 5.0	Equilateral triangle	EP1
A = 5.0, B = 5.0, C = 3.0	Isosceles triangle	EP2
A = 3.0, B = 4.0, C = 5.0	Scalene triangle	EP3
A = 3.0, B = 4.0, C = 5.0	Right-angled triangle	EP4
A = 1.0, B = 2.0, C = 3.0	Non-triangle	EP5
A = -1.0, B = 2.0, C = 3.0	Non-positive input	EP6

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

To verify the boundary for the scalene triangle condition A+B>C

Tester Action and Input Data	Expected Outcome	Equivalence Class
A = 2.0, B = 3.0, C = 4.0	Scalene triangle	A+B=C−ϵ
A = 3.0, B = 4.0, C = 5.0	Scalene triangle	A+B=C (valid)
A = 3.0, B = 5.0, C = 7.0	Scalene triangle	A+B=C+ϵ

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

To verify the boundary for the isosceles triangle condition A=C

Tester Action and Input Data	Expected Outcome	Equivalence Class
A = 3.0, B = 4.0, C = 3.0	Isosceles triangle	A=C
A = 4.0, B = 4.0, C = 5.0	Isosceles triangle	A=C+ ϵ

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

To verify the boundary for the equilateral triangle condition A=B=C

Tester Action and Input Data	Expected Outcome	Equivalence Class
A = 4.0, B = 4.0, C = 4.0	Isosceles triangle	A=C=B

A = 5.0, B = 5.0, C = 5.0	Isosceles triangle	A=C=B

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

To verify the boundary for the right-angled triangle condition $A^2 + B^2 = C^2$

Tester Action and Input Data	Expected Outcome	Equivalence Class
A = 3.0, B = 4.0, C = 5.0	Right-angled triangle	$A^2 + B^2 = C^2$
A = 5.0, B = 12.0, C = 13.0	Right-angled triangle	$A^2 + B^2 = C^2$

g) Non-Triangle Case Test Points

To explore the boundaries of non-triangle cases:

Tester Action and Input Data	Expected Outcome	Equivalence Class
A = 1.0, B = 2.0, C = 3.0	Non-triangle	$A + B = C$
A = 2.0, B = 2.0, C = 5.0	Non-triangle	$A + B < C$

h) Non-Positive Input Test Points

To explore cases with non-positive lengths:

Tester Action and Input Data	Expected Outcome	Equivalence Class
A = 0.0, B = 1.0, C = -1.0	Non-Positive input	$A = 0$
A = -1.0, B = 1.0, C = 1.0	Non-Positive input	$A < 0$