Software Testing Lab Report

Functional Testing (Black-Box Testing)

Name: Anshu Dhankecha

ID: 202201234

Group: G3

Q1. Calendar Date Program Testing

Consider a program that calculates the previous date given an input of day, month, and year. Input ranges:

- Month: 1-12
- Day: 1-31
- Year: 1900-2015

Test Cases Using Equivalence Partitioning

Test Case ID	Input (D,M,Y)	Expected Output	Description
EP1	12, 8, 2002	11, 8, 2002	Valid mid-month date
EP2	1, 3, 2001	28, 2, 2001	Month beginning
EP3	28, 2, 2008	27, 2, 2008	Leap year February
EP4	31, 4, 2000	Invalid Date	Invalid day for month
EP5	0, 6, 2010	Invalid Date	Invalid day value
EP6	1, 5, 2003	30, 4, 2003	Month transition

Boundary Value Analysis Test Cases

Test Case ID	Input (D,M,Y)	Expected Output	Description
BV1	1, 1, 1900	31, 12, 1899	Lower boundary year
BV2	12, 2015	30, 12, 2015	Upper boundary year
BV3	35, 3, 2005	Invalid Date	Beyond max days
BV4	1, 13, 2010	Invalid Date	Beyond max month

Q2. Array Operations Testing

P1. Linear Search Function

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

Test Cases

Test Case ID	Input Array	Search Value	Expected Output	Category
LS1	{4, 8, 2, 9, 1}	8	1	EP
LS2	{3, 7, 1, 9, 4}	5	-1	EP
LS3	{6}	6	0	BV
LS4		3	-1	BV

P2. Count Items Function

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

Test Cases

Test Case ID	Input Array	Search Value	Expected Output	Category
CI 1	{4, 2, 4, 1, 4}	4	3	EP
CI 2	{1, 3, 5, 7, 9}	2	0	EP
CI 3	{8, 8, 8, 8}	8	4	BV
CI 4		5	0	BV

P3. Binary Search Function

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

Test Cases

Test Case ID	Input Array	Search Value	Expected Output	Category
BS1	{2, 4, 6, 8, 10}	6	2	EP
BS2	{1, 3, 5, 7, 9}	4	-1	EP
BS3	{5}	5	0	BV
BS4		2	-1	BV

P4. Triangle Classification

```
int classifyTriangle(int a, int b, int c) {
    if(a <= 0 || b <= 0 || c <= 0 || a >= b + c || b >= a + c || c >= a + b)
        return 3; // Invalid
    if(a == b && b == c)
        return 0; // Equilateral
    if(a == b || b == c || a == c)
        return 1; // Isosceles
    return 2; // Scalene
}
```

Test Cases

Test Case ID	Input (a,b,c)	Expected Output	Category
TR1	(5, 5, 5)	0	EP
TR2	(4, 4, 6)	1	EP
TR3	(3, 4, 5)	2	EP
TR4	(1, 1, 3)	3	EP
TR5	(0, 5, 5)	3	BV
TR6	(2, 2, 4)	3	BV

P5. String Prefix Function

```
bool isPrefix(string s1, string s2) {
    if(s1.length() > s2.length()) return false;
    for(int i = 0; i < s1.length(); i++) {
        if(s1[i] != s2[i]) return false;
    }
    return true;
}
```

Test Cases

Test Case ID	Input (s1, s2)	Expected Output	Category
SP1	"pro", "program"	true	EP
SP2	"test", "testing"	true	EP
SP3	"java", "python"	false	EP
SP4	"", "code"	true "long",	BV
SP5	"short"	false	BV

P6. Enhanced Triangle Classification

Equivalence Classes:

1. Valid Triangles:
    - Equilateral: all sides equal
    - Isosceles: exactly two sides equal
    - Scalene: no sides equal
    - Right-angled: satisfies Pythagorean theorem
2. Invalid Triangles:
    - Sides violate triangle inequality
    - Non-positive sides

Test Cases:

Test Case ID	Input (a,b,c)	Expected Output	Test Category
ET1	(4.0, 4.0, 4.0)	Equilateral	Valid Valid Valid
ET2	(5.0, 5.0, 7.0)	Isosceles	Valid Triangle
ET3	(3.0, 4.0, 5.0)	Right-angled	Inequality Non-
ET4	(2.0, 3.0, 4.0)	Scalene	positive
ET5	(1.0, 1.0, 3.0)	Invalid	
ET6	(0.0, 4.0, 5.0)	Invalid	

Test Case ID	Input (a,b,c)	Expected Output	Test Category
ET7	(6.0, 8.0, 10.0)	Right-angled	Boundary
ET8	(7.0, 7.0, 7.0)	Equilateral	Boundary

Note: All test cases have been verified with the implemented functions and produce the expected outputs.