Software Testing Lab Report

Functional Testing (Black-Box Testing)

Name: Harsh Rathod

ID: 202201212

Group: G3

Q1. Previous Date Calculator Testing

Consider a program for determining the previous date with input ranges:

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

Equivalence Partitioning Test Cases

Test Case ID	Input (D,M,Y)	Expected Output	Description
EP1	18, 7, 2005	17, 7, 2005	Mid-month date
EP2	1, 6, 2003	31, 5, 2003	Month transition
EP3	29, 2, 2012	28, 2, 2012	Leap year case
EP4	31, 9, 2000	Invalid Date	Invalid September date
EP5	-1, 8, 2010	Invalid Date	Negative day
EP6	1, 1, 2004	31, 12, 2003	Year transition

Boundary Value Analysis Test Cases

Test Case ID	Input (D,M,Y)	Expected Output	Description
BV1	1, 1, 1900	31, 12, 1899	Minimum year boundary
BV2	31, 12, 2015	30, 12, 2015	Maximum year boundary
BV3	32, 7, 2008	Invalid Date	Day overflow
BV4	15, 0, 2010	Invalid Date	Invalid 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	{7, 3, 9, 2, 6}	9	2	EP
LS2	{5, 8, 1, 4, 7}	0	-1	EP
LS3	{4}	4	0	BV
LS4		5	-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
CI1	{3, 7, 3, 8, 3}	3	3	EP
CI2	{2, 4, 6, 8, 10}	5	0	EP
CI3	{5, 5, 5, 5}	5	4	BV
CI4		2	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	{1, 3, 5, 7, 9}	5	2	EP
BS2	{2, 4, 6, 8, 10}	7	-1	EP
BS3	{3}	3	0	BV
BS4		1	-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	(6, 6, 6)	0	EP
TR2	(5, 5, 7)	1	EP
TR3	(4, 5, 6)	2	EP
TR4	(2, 2, 5)	3	EP
TR5	(-1, 4, 4)	3	BV
TR6	(1, 1, 2)	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	"dev", "develop"	true	EP
SP2	"code", "coding"	true	EP
SP3	"soft", "hard"	false	EP
SP4	"", "test"	true	BV
SP5	"python", "py"	false	BV

P6. Enhanced Triangle Classification

Equivalence Classes:

1. Valid Triangles:
    - Equilateral: all sides equal
    - Isosceles: two sides equal
    - Scalene: no sides equal
    - Right-angled: follows Pythagorean theorem
2. Invalid Cases:
    - Triangle inequality violation
    - Zero or negative sides

Test Cases:

Test Case ID	**Input (a,b,c)**	**Expected Output**	**Test Category**
ET1	(5.0, 5.0, 5.0)	Equilateral	Valid
ET2	(6.0, 6.0, 8.0)	Isosceles	Valid
ET3	(5.0, 12.0, 13.0)	Right-angled	Valid
ET4	(3.0, 4.0, 6.0)	Scalene	Valid
ET5	(2.0, 2.0, 5.0)	Invalid	Triangle Inequality
ET6	(-2.0, 4.0, 4.0)	Invalid	Non-positive

Test Case ID	Input (a,b,c)	Expected Output	Test Category
ET7	(8.0, 15.0, 17.0)	Right-angled	Boundary
ET8	(10.0, 10.0, 10.0)	Equilateral	Boundary

Note: All test cases have been verified against the implemented functions to ensure correct expected outputs.