



IT314 Lab 8

Functional Testing (Black-Box)

Bhavya Patel
202201179

Q.1. Consider a program for determining the previous date. Its input is triple of day, month and year with the following ranges $1 \leq \text{month} \leq 12$, $1 \leq \text{day} \leq 31$, $1900 \leq \text{year} \leq 2015$. The possible output dates would be previous date or invalid date. Design the equivalence class test cases?

Equivalence class

1. Valid input for a previous date:

- Equivalence class 1: Day, month, and year that corresponds to a valid date that is not the lowest boundary (day = 2, month = 3, year = 2010).
- Equivalence class 2: Day = 1, month > 1 (day = 1, month = 4, year = 2005).
- Equivalence class 3: Day > 1, month = 1 (day = 15, month = 1, year = 1999).

2. Valid input for the lowest boundary:

- Equivalence class 4: Day = 1, month = 1 (day = 1, month = 1, year = 1900).

3. Valid input for the highest boundary:

- Equivalence class 5: Day = 31, month = 12, year = 2015 (day = 31, month = 12, year = 2015).

4. Invalid inputs:

- Equivalence class 6: Day < 1 (day = 0, month = 6, year = 2000).
- Equivalence class 7: Day > 31 (day = 32, month = 7, year = 1995).
- Equivalence class 8: Month < 1 (day = 20, month = 0, year = 2012).
- Equivalence class 9: Month > 12 (day = 10, month = 13, year = 2008).
- Equivalence class 10: Year < 1900 (day = 5, month = 9, year = 1899).
- Equivalence class 11: Year > 2015 (day = 3, month = 11, year = 2016).

5. Special cases:

- Equivalence class 12: Leap year (day = 29, month = 2, year = 2000).
- Equivalence class 13: Non-leap year (day = 29, month = 2, year = 1900).

Equivalence Class Test Cases

Test ID	Tester Action and Input Data (Day, Month, Year)	Expected Outcome	Equivalence Partitioning / Boundary Value Analysis
TC1	Day = 2, Month = 3, Year = 2010	Previous Date: 01/03/2010	EP1: Valid date (Day > 1, Month > 1, Year in range)
TC2	Day = 1, Month = 4, Year = 2005	Previous Date: 31/03/2005	EP2: Valid (First day of month, Month > 1)
TC3	Day = 15, Month = 1, Year = 1999	Previous Date: 14/01/1999	EP3: Valid (Day > 1, Month = 1)
TC4	Day = 1, Month = 1, Year = 1900	Previous Date: Invalid (No previous date before 01/01/1900)	EP4: Valid lowest boundary (Day = 1, Month = 1, Year = 1900)
TC5	Day = 31, Month = 12, Year = 2015	Previous Date: 30/12/2015	EP5: Valid highest boundary (Day = 31, Month = 12, Year = 2015)
TC6	Day = 0, Month = 6, Year = 2000	Error: Invalid date	EP6: Invalid (Day < 1)
TC7	Day = 32, Month = 7, Year = 1995	Error: Invalid date	EP7: Invalid (Day > 31)
TC8	Day = 20, Month = 0, Year = 2012	Error: Invalid date	EP8: Invalid (Month < 1)
TC9	Day = 10, Month = 13, Year = 2008	Error: Invalid date	EP9: Invalid (Month > 12)
TC10	Day = 5, Month = 9, Year = 1899	Error: Invalid date	EP10: Invalid (Year < 1900)
TC11	Day = 3, Month = 11, Year = 2016	Error: Invalid date	EP11: Invalid (Year > 2015)
TC12	Day = 29, Month = 2, Year = 2000	Previous Date: 28/02/2000	EP12: Leap year valid (Day = 29, Month = 2, Year = Leap Year)
TC13	Day = 29, Month = 2, Year = 1900	Error: Invalid date	EP13: Non-leap year (Feb 29 does not exist in 1900)
TC14	Day = 1, Month = 2, Year = 1900	Previous Date: 31/01/1900	BVA: Boundary value at Day = 1 and Month > 1
TC15	Day = 1, Month = 1, Year = 2015	Previous Date: 31/12/2014	BVA: Boundary value at Day = 1 and Year > 1900

TC16	Day = 1, Month = 1, Year = 1901	Previous Date: 31/12/1900	BVA: Boundary value at Day = 1, Month = 1, and Year = 1901
------	------------------------------------	------------------------------	--

Q.2. Programs:

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.

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

P1. linearSearch(v, a[])

Description: This function searches for the first occurrence of a value 'v' in the array 'a'. It returns the index of the first occurrence or -1 if the value is not found.

Equivalence Classes (EP):

- EP1: v is present in the array a.
- EP2: v is not present in the array a.
- EP3: Empty array (a.length = 0).

Boundary Value Analysis (BVA):

- BVA1: v is the first element of the array (a[0]).
- BVA2: v is the last element of the array (a[length-1]).
- BVA3: Array with one element (a.length = 1).

P1: linearSearch(v, a[])

Test ID	Tester Action and Input Data	Expected Outcome	Equivalence Partitioning / Boundary Value Analysis
TC1	Search for 5 in [1, 2, 3, 5]	3	EP1 (v is present)
TC2	Search for 4 in [1, 2, 3, 5]	-1	EP2 (v is not present)
TC3	Search for 1 in []	-1	EP3 (Empty array)
TC4	Search for 1 in [1]	0	BVA3 (Array with one element)
TC5	Search for 5 in [5]	0	BVA1 (v is the first element)
TC6	Search for 1 in [5]	-1	EP2 (v is not present)
TC7	Search for 5 in [5, 1, 3]	0	BVA1 (v is the first element)
TC8	Search for 3 in [5, 1, 3]	2	EP1 (v is present)

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

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

P2. countItem(v, a[])

Description: This function returns the number of times a value 'v' appears in an array 'a'.

Equivalence Classes (EP):

- EP1: v appears in the array a multiple times.
- EP2: v appears in the array a once.
- EP3: v does not appear in the array a.
- EP4: Empty array (a.length = 0).

Boundary Value Analysis (BVA):

- BVA1: v appears at the start of the array.
- BVA2: v appears at the end of the array.
- BVA3: Array with one element.

P2: countItem(v, a[])

Test ID	Tester Action and Input Data	Expected Outcome	Equivalence Partitioning / Boundary Value Analysis
TC1	Count 2 in [1, 2, 2, 3]	2	EP1 (v appears multiple times)
TC2	Count 3 in [1, 2, 3]	1	EP2 (v appears once)
TC3	Count 4 in [1, 2, 3]	0	EP3 (v does not appear)
TC4	Count 2 in []	0	EP4 (Empty array)
TC5	Count 1 in [1]	1	BVA3 (Array with one element)
TC6	Count 1 in [1, 2, 3, 1]	2	EP1 (v appears multiple times)
TC7	Count 2 in [2]	1	EP2 (v appears once)
TC8	Count 2 in [2, 2, 3]	2	EP1 (v appears multiple times)

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.

```
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);
}
```

P3. `binarySearch(v, a[])`

Description: This function performs a binary search on a sorted array of integers 'a' for the value 'v'. Returns an index of 'v' or -1 if not found.

Equivalence Classes (EP):

- EP1: v is present in the array a.
- EP2: v is not present in the array a.
- EP3: Empty array (a.length = 0).

Boundary Value Analysis (BVA):

- BVA1: v is the middle element of the array.
- BVA2: v is the first element of the array.
- BVA3: v is the last element of the array.
- BVA4: Array with one element (a.length = 1).

P3: `binarySearch(v, a[])`

Test ID	Tester Action and Input Data	Expected Outcome	Equivalence Partitioning / Boundary Value Analysis
TC1	Search for 5 in [1, 2, 3, 5]	3	EP1 (v is present)
TC2	Search for 4 in [1, 2, 3, 5]	-1	EP2 (v is not present)
TC3	Search for 2 in []	-1	EP3 (Empty array)
TC4	Search for 3 in [3]	0	BVA4 (Array with one element)
TC5	Search for 1 in [1, 2, 3, 4]	0	BVA2 (v is the first element)
TC6	Search for 4 in [1, 2, 3, 4]	3	BVA3 (v is the last element)
TC7	Search for 2 in [1, 2, 3]	1	BVA1 (v is the middle element)
TC8	Search for 5 in [1, 2, 3, 4]	-1	EP2 (v is not present)

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).

```
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);  
}
```

P4. `triangle(a, b, c)`

Description: This function takes three integer parameters (a, b, c) representing the lengths of the sides of a triangle. It returns whether the triangle is equilateral, isosceles, scalene, or invalid.

Equivalence Classes (EP):

- EP1: Equilateral triangle ($a = b = c$).
- EP2: Isosceles triangle ($a = b, a \neq c$).
- EP3: Scalene triangle ($a \neq b \neq c$).
- EP4: Invalid triangle ($a + b \leq c$ or $a + c \leq b$ or $b + c \leq a$).

Boundary Value Analysis (BVA):

- BVA1: $a + b = c$ (Invalid boundary case).
- BVA2: $a = b = c = 1$ (Smallest valid triangle).

- BVA3: $a + b > c$ (Valid boundary case).

P4: triangle(a, b, c)

Test ID	Tester Action and Input Data	Expected Outcome	Equivalence Partitioning / Boundary Value Analysis
TC1	Check triangle with sides (3, 3, 3)	Equilateral	EP1 (Equilateral triangle)
TC2	Check triangle with sides (3, 3, 4)	Isosceles	EP2 (Isosceles triangle)
TC3	Check triangle with sides (3, 4, 5)	Scalene	EP3 (Scalene triangle)
TC4	Check triangle with sides (1, 2, 3)	Invalid	EP4 (Invalid triangle)
TC5	Check triangle with sides (1, 1, 1)	Equilateral	BVA2 (Smallest valid triangle)
TC6	Check triangle with sides (1, 2, 2)	Isosceles	EP2 (Isosceles triangle)
TC7	Check triangle with sides (5, 3, 4)	Scalene	EP3 (Scalene triangle)
TC8	Check triangle with sides (1, 2, 3)	Invalid	EP4 (Invalid triangle)

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).

```
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;
}
```

P5. `prefix(s1, s2)`

Description: This function returns whether or not the string `s1` is a prefix of string `s2`.

Equivalence Classes (EP):

- EP1: `s1` is a prefix of `s2`.
- EP2: `s1` is not a prefix of `s2`.
- EP3: `s1` has the same length as `s2` but is not a prefix.
- EP4: `s1` has a greater length than `s2`.

Boundary Value Analysis (BVA):

- BVA1: `s1` is an empty string.
- BVA2: `s1` equals `s2` (both strings are the same).
- BVA3: `s1` is one character less than `s2`.

P5: prefix(s1, s2)

Test ID	Tester Action and Input Data	Expected Outcome	Equivalence Partitioning / Boundary Value Analysis
TC1	Check if "abc" is prefix of "abcd"	true	EP1 (s1 is a prefix of s2)
TC2	Check if "def" is prefix of "abcd"	false	EP2 (s1 is not a prefix of s2)
TC3	Check if "abc" is prefix of "abc"	true	EP3 (s1 equals s2)
TC4	Check if "abcd" is prefix of "abc"	false	EP4 (s1 is longer than s2)
TC5	Check if "" is prefix of "abc"	true	BVA1 (s1 is an empty string)
TC6	Check if "ab" is prefix of "abc"	true	BVA3 (s1 is one character less than s2)
TC7	Check if "a" is prefix of "abc"	true	EP1 (s1 is a prefix of s2)
TC8	Check if "abc" is prefix of "ab"	false	EP4 (s1 is longer than s2)



Answer to P6

a) Equivalence Classes:

- (a) Equilateral Triangle: All sides are equal ($A = B = C$).
- (b) Isosceles Triangle: Two sides are equal, and the third is different ($A = B \neq C$, $A \neq B = C$, $A = C \neq B$).
- (c) Scalene Triangle: All sides are different ($A \neq B \neq C$).
- (d) Right-Angled Triangle: Satisfies the Pythagorean theorem ($A^2 + B^2 = C^2$).
- (e) Non-Triangle: It cannot form a triangle ($A + B \leq C$, $B + C \leq A$, $C + A \leq B$).

b) Extensive Test Cases:

(a) Equivalence Class: Equilateral Triangle

- Test Case 1: $A = 1$, $B = 1$, $C = 1$ (Minimum positive values)
- Test Case 2: $A = 10$, $B = 10$, $C = 10$ (Larger positive values)

(b) Equivalence Class: Isosceles Triangle

- Test Case 3: $A = 3$, $B = 3$, $C = 4$ ($A = B \neq C$)
- Test Case 4: $A = 4$, $B = 3$, $C = 3$ ($A \neq B = C$)
- Test Case 5: $A = 3$, $B = 4$, $C = 3$ ($A = C \neq B$)

(c) Equivalence Class: Scalene Triangle

- Test Case 6: $A = 3$, $B = 4$, $C = 5$ (Regular scalene triangle)
- Test Case 7: $A = 1$, $B = 2$, $C = 3$ (Smallest positive values)

(d) Equivalence Class: Right-Angled Triangle

- Test Case 8: $A = 3$, $B = 4$, $C = 5$ ($A^2 + B^2 = 9 + 16 = 25 = C^2$)
- Test Case 9: $A = 5$, $B = 12$, $C = 13$ (Another right-angled triangle)

(e) Equivalence Class: Non-Triangle

- Test Case 10: $A = 1$, $B = 2$, $C = 6$ ($A + B = 3 < C$)
- Test Case 11: $A = 0$, $B = 0$, $C = 0$ (All sides are zero)
- Test Case 12: $A = 1$, $B = 1$, $C = 2$ ($A + B = 2 = C$)

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

- (a) Test Case 13: $A = 3$, $B = 4$, $C = 6$ ($A + B = 7 > C$)
- (b) Test Case 14: $A = 1$, $B = 1$, $C = 2$ ($A + B = 2 < C$)

d) Boundary Condition $A = C$ (Isosceles Triangle):

- (a) Test Case 15: $A = 5$, $B = 4$, $C = 5$ ($A = C$)

- (b) Test Case 16: $A = 1, B = 1, C = 2$ ($A \neq C$)
- e) Boundary Condition $A = B = C$ (Equilateral Triangle):
 - (a) Test Case 17: $A = 4, B = 4, C = 4$ ($A = B = C$)
 - (b) Test Case 18: $A = 1, B = 2, C = 3$ ($A \neq B \neq C$)
- f) Boundary Condition $A^2 + B^2 = C^2$ (Right-Angled Triangle):
 - (a) Test Case 19: $A = 3, B = 4, C = 5$ ($A^2 + B^2 = 9 + 16 = 25 = C^2$)
 - (b) Test Case 20: $A = 7, B = 24, C = 25$ (Another right-angled triangle) or Non-Triangle Case (Boundary Exploration):
 - (c) Test Case 21: $A = 1, B = 2, C = 3$ ($A + B = 3 < C$)
 - (d) Test Case 22: $A = 0, B = 0, C = 1$ (A and B are zero, $A + B = 0 < C$)
 - (e) Test Case 23: $A = 1, B = 1, C = 3$ ($A + B = 2 < C$)
- g) For Non-Positive Input (Boundary Exploration):
 - (a) Test Case 24: $A = -1, B = 2, C = 3$ (A is non-positive)
 - (b) Test Case 25: $A = 1, B = -2, C = 3$ (B is non-positive)
 - (c) Test Case 26: $A = 1, B = 2, C = -3$ (C is non-positive)
 - (d) Test Case 27: $A = 0, B = 2, C = 3$ (A is zero)
 - (e) Test Case 28: $A = 1, B = 0, C = 3$ (B is zero)
 - (f) Test Case 29: $A = 1, B = 2, C = 0$ (C is zero)