IT-314
Software Engineering
LAB - 08
Software Testing
Lab Session - Functional Testing (Black-Box)

Student Id: 202201015
Name: Meghavi Gohil

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

Equivalence Class:
E1: 1<=day<=31 (**valid**)
E2: day<1(**invalid**)
E3: day>31 (**invalid**)
E4: 1<=month<=12(**valid**)
E5: month<1(**invalid**)
E6:month>12(**invalid**)

E7: 1900<=year<=2015(**valid**)
E8: year<1900(**invalid**)
E9: year>2015(**invalid**)

Equivalence Partitioning (EP) Test Cases

Input Data	Class Covered	Expected Outcome
EP1: (2, 1, 1900)	E1,E4,E7	Previous day: (1, 1, 1900)
EP2: (1, 3, 2010)	E1,E4,E7	Previous day: (28, 2, 2010)
EP3: (1, 5, 2000)	E1,E4,E7	Previous day: (30, 4, 2000)
EP4: (29, 2, 2012)	E1,E4,E7	Previous day: (28, 2, 2012)
EP5: (32, 2, 2011)	E3	invalid date
EP6: (3, 1, 2016)	E9	invalid year
EP7: (15, 13, 2010)	E6	invalid month
EP8: (15, 6, 1899)	E8	invalid year

Boundary Value Analysis (BVA) Test Cases

Input Data	Expected Outcome
BVA1: (1, 1, 1900)	Previous day: no previous day for this input
BVA2: (2, 1, 1900)	Previous day: (1, 1, 1900)
BVA3: (31, 12, 2015)	Previous day: (30, 12, 2015)
BVA4: (1, 3, 2016)	Invalid year
BVA5: (29, 2, 2012)	Previous day: (28, 2, 2012)
BVA6: (30, 4, 2010)	Previous day: (29, 4, 2010)
BVA7: (0, 3, 2010)	invalid day
BVA8: (312, 4, 2010)	invalid day

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);
}
```

This program implements a linear search algorithm for an array of integers.

- a) Equivalence Classes and Boundary Value Classes:
    1. v is present in the array.
    2. v is not present in the array.
    3. The array is empty.
    4. v present at 0th index
    5. v present a last index
- b) Test Cases:
    1. v = 5, array = [1, 2, 5, 7] (v is present(E1)).
    2. v = 9, array = [1, 2, 5, 7] (v is not present(E2)).
    3. v = 1, array = [] (array is empty(E3)).
    4. v = 2 , array = [2,1,3,5] (E4)
    5. v = 5, array = [2 , 2, 4 ,5] (E5)

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);
```

This function counts the number of occurrences of a value in an array.

- a) Equivalence Classes and Boundary Value Classes:
    1. 0<count(v)<n occurs multiple times.
    2. count(v)=0.
    3. count(v)=n
    4. The array is empty.
- b) Test Cases:
    1. v = 2, array = [2, 2, 2, 3, 4] (E1).
    2. v = 3, array = [1, 2, 2] (E2).
    3. v = 2, array = [2, 2, 2] (E3).
    4. v = 1, array = [] (E4).

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);
}
```

- a) Equivalence Classes and Boundary value classes:
    1. v is present in the array.
    2. v is not present in the array.
    3. The array has only one element.
    4. The array is empty.
- b) Test Cases:
    1. v = 4, array = [1, 2, 3, 4, 5, 6] (E1).
    2. v = 7, array = [1, 2, 3, 4, 5, 6] (E2).
    3. v = 2, array = [2] (E3).
    4. v = 1, array = [] (E4).

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);
}
```

- a) Equivalence Classes:
    1. Equilateral triangle (a == b == c).
    2. Isosceles triangle (a == b || a == c || b == c).
    3. Scalene triangle (a ≠ b ≠ c).
    4. Invalid triangle (sum of two sides ≤ the third).
    5. Non-positive lengths (one or more sides ≤ 0).
- b) Test Cases:
    1. a = 3, b = 3, c = 3 (E1).
    2. a = 3, b = 3, c = 5 (E2).
    3. a = 3, b = 4, c = 5 (E3).
    4. a = 1, b = 2, c = 3 (E4).
    5. a = 0, b = 2, c = 3 (E5).

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;
}
```

- a) Equivalence Classes:
    1. s1 is a prefix of s2.
    2. s1 is not a prefix of s2.
    3. s1 is longer than s2.
- b) Test Cases:
    1. s1 = "abc", s2 = "abcdef" (E1).
    2. s1 = "xyz", s2 = "abcdef" (E2).
    3. s1 = "abcdefg", s2 = "abc" (E3).

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 the equivalence classes for the system
b) Identify test cases to cover the identified equivalence classes. Also, explicitly mention which test case would cover which equivalence class. (Hint: you must need to be ensure that the identified set of test cases cover all identified equivalence classes)
c) For the boundary condition A + B > C case (scalene triangle), identify test cases to verify the boundary.
d) For the boundary condition A = C case (isosceles triangle), identify test cases to verify the boundary.

e) For the boundary condition A = B = C case (equilateral triangle), identify test cases to verify the boundary.
f) For the boundary condition A2 + B2 = C2 case (right-angle triangle), identify test cases to verify the boundary.
g) For the non-triangle case, identify test cases to explore the boundary.
h) For non-positive input, identify test points.

(a) Equivalence Classes:

1. Equilateral: All sides are equal (A == B == C).(Valid)
2. Isosceles: Two sides are equal (A == B || A == C || B == C).(Valid)
3. Scalene: All sides are different (A ≠ B ≠ C).(Valid)
4. Right-angled: Pythagorean theorem is satisfied ($A^2 + B^2 = C^2$).(Valid)
5. The sum of any two sides is less than or equal to the third side (A + B <= C).(Invalid)
6. One or more sides have zero or negative lengths.(Invalid)

b) Test Cases:

1. A = 3.0, B = 3.0, C = 3.0 (E1).**Valid**
2. A = 4.0, B = 4.0, C = 5.0 (E2).**Valid**
3. A = 3.0, B = 4.0, C = 5.0 (E4).**Valid**
4. A = 5.0, B = 7.0, C = 10.0 (E3).**Valid**
5. A = 1.0, B = 1.0, C = 3.0 (E5).**Invalid**
6. A = -1.0, B = 2.0, C = 3.0 (E6).**Invalid**

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

1. A = 2.0, B = 3.0, C = 4.999 (just valid).
2. A = 2.0, B = 3.0, C = 5.0 (boundary case).
3. A = 2.0, B = 3.0, C = 5.001 (invalid).

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

1. A = 4.0, B = 5.0, C = 4.0001 (just valid).
2. A = 4.0, B = 5.0, C = 4.0 (boundary case).

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

1. A = 3.0, B = 3.0, C = 3.00001 (just valid).

2. A = 3.0, B = 3.0, C = 3.0 (boundary case).

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

1. A = 3.0, B = 4.0, C = 5.0 (valid right-angled triangle).
2. A = 1.0, B = 1.0, C = 1.414 (boundary).

g) Non-triangle Case:

1. A = 1.0, B = 2.0, C = 3.0 (invalid triangle).

h) Non-positive Input:

1. A = -3.0, B = 2.0, C = 4.0 (negative side).
2. A = 0, B = 2.0, C = 4.0 (zero side).