IT314 Software Engineering LAB - 08

Name :: Bhara Chintan

ID :: 202201060

Que 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?

Write a set of test cases (i.e., test suite) – specific set of data – to properly test the programs. Your test suite should include both correct and incorrect inputs.

1. Enlist which set of test cases have been identified using Equivalence Partitioning and Boundary Value Analysis separately.
2. Modify your programs such that it runs, and then execute your test suites on the program. While executing your input data in a program, check whether the identified expected outcome (mentioned by you) is correct or not.

Solution ::

Equivalence Partitioning :

Date	Month	Year	Equivalence Class	Expected Output
10	02	1973	EV1	True(All fields are in valid range)
13	13	1993	EV2	Error Message(Invalid Month)

32	10	2004	EV3	Error Message(Invalid Date)
10	07	2063	EV4	Error Message(Out of bound Year)

Boundary Value Analysis :

Date	Month	Year	Boundary Class	Expected Output
31*	12	2015	BV1	True(All fields on maximum value)
1	1	1900	BV1	True(All fields on minimum value)
0	12	2004	BV2	Error Message(Date is out of bound)
1	0	2009	BV3	Error Message(Month is out of bound)
16	7	2016	BV4	Error Message(year is out of bound)

Note (*) : - Here according to different conditions date limit can changes, as given below,

- January,March,May,July,August,October,December have 31 days.
- April,June,September,November contain 30 days so their maximum is 30 only.
- February again has 2 cases ; if leap year then it contains at maximum 29 days and if not then only 28 days.

Que 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;
}
```

Equivalence Partitioning :

Integer V	Integer array	Equivalence class	Expected outcome
5	[1,2,3,4,5]	EV1	True(returns index)
6	[1,2,3,4,5]	EV2	True(returns -1)
10^{12} + 7	[Random array]	EV3	Error(Overflow V)
5	[1,2,...10^9+7]	EV4	Error(Overflow array length)

Boundary Value Analysis :

Integer V	Integer array	Boundary class	Expected outcome
2^{31}	[Random Array]	BV1	True(V At the boundary)
2^{-31}	[Random Array]	BV1	True(V At the boundary)
2	[1,3,5,...,2^{31}]	BV1	True(Array length At the boundary)
2	[2^{-31},...,-3,-1]	BV1	True(Array length At the boundary)
2^{31} + 1	[Random Array]	BV2	Error(Out of bound V)
2^{-31} - 1	[Random Array]	BV2	Error(Out of bound V)
2	[1,3,5,...,2^{31}+1]	BV3	Error(Out of bound Array)
2	[2^{-31}-1,...,-3,-1]	BV3	Error(Out of bound Array)

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);
}
```

Equivalence Value Analysis :

Integer V	Integer array	Equivalence class	Expected outcome
5	[1,2,3,4,5]	EV1	True(returns count)
6	[1,2,3,4,5]	EV2	True(returns count)
10^{12} + 7	[Random array]	EV3	Error(Overflow V)
5	[0,2,4,...10^9+8]	EV4	Error(Overflow array length)

Boundary Value Analysis :

Integer V	Integer array	Boundary class	Expected outcome
2^{31}	[Random Array]	BV1	True(V At the boundary and return)
2^{-31}	[Random Array]	BV1	True(V At the boundary and return)
2	[1,3,5,...,2^{31}]	BV1	True(A length At the boundary and return)
2	[2^{-31},...,-3,-1]	BV1	True(A length At the boundary and return)
2^{31} + 1	[Random Array]	BV2	Error(Out of bound V)
2^{-31} - 1	[Random Array]	BV2	Error(Out of bound V)
2	[1,3,5,...,2^{31}+1]	BV3	Error(Out of bound Array)
2	[2^{-31}-1,...,-3,-1]	BV3	Error(Out of bound Array)

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.

Assumption: the elements in the array a are sorted in non-decreasing order.

```
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;
}
```

Example array : [1,3,5,7,8,9]

Equivalence Value analysis :

Entered Value	Expected Out
5	2
1	0
9	5
6	-1

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);
}
```

E1: all sides equal

E2: any two sides are equal

E3: all sides not equal but valid triangle

E4: a>=b+c

E5: b>=a+c

E6: c>=a+b

Input (a,b,c)	Expected Out
2,2,2	Equilateral
1,2,1	Isosceles
3,2,4	Scalene
3,1,2	Invalid
2,4,5	Invalid
2,1,4	Invalid

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;
}
```

Entered String	Expected Out
ab	false
abc	true
c	true
abed	false

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.

Equivalence Class :

E1: positive sides(valid)

E2: any side <=0 (invalid)

E3: valid triangle(valid)

E4: invalid triangle(invalid)

E5: a=b=c (valid)

E6: two sides are equal (valid)

E7: all sides are different length (valid)

E8: Right angle triangle

Test Case :

No	Input (a,b,c)	Expected Output	Class Covered
1	2,-1,3	Invalid	E2
2	2,2,2	Invalid	E2
3	1,2,1	Equilateral	E1,E3,E5
4	3,2,4	Isosceles	E1,E3,E6
5	2,4,5	Invalid	E1,E4
6	2,3,5	Right angle	E1,E3,E8

c) For the boundary condition A + B > C case (scalene triangle), identify test cases to verify the boundary.

7	1,2,3	Invalid	E1,E4
8	3,4,6	Scalene	E1,E4,E7

d) For the boundary condition A = C case (isosceles triangle), identify test cases to verify the boundary.

9	5,6,5	Isosceles	E1,E3,E6

10	5,7,5	Isosceles	E1,E3,E6

e) For the boundary condition A = B = C case (equilateral triangle), identify test cases to verify the boundary.

11	5,5,5	Equilateral	E1,E3,E5

f) For the boundary condition A2 + B2 = C2 case (right-angle triangle), identify test cases to verify the boundary.

12	3,4,5	Right Angle Triangle	E1,E3,E8

g) For the non-triangle case, identify test cases to explore the boundary.

13	1,2,3	Invalid	E1,E4