Dhirubhai Ambani Institute of Information and Communication Technology

IT314: Software Engineering

LAB-8

NAME - Nishil Patel

STUDENT ID - 202201166

SOFTWARE TESTING

Question-1

1) Equivalence Class Partitioning (EP):

Equivalence partitioning divides the input domain into several classes that are expected to behave similarly

Valid Equivalence Classes:

- Day: $(1 \leq day \leq 31)$

- Month: (1 ≤ month ≤ 12)
- Year: (1900 ≤ year ≤ 2015)

Invalid Equivalence Classes:

- Day<1 or Day>31
- Month <1 or Month > 12
- Year < 1900 or Year > 2015

2. Boundary Value Analysis (BV):

BVA tests values at the boundaries between partitions.

Boundaries for Day:

Minimum value:1

Maximum value:31

Boundaries for Month:

Minimum value:1

Maximum value:12

Boundaries for Year:

Minimum Value:1900

Maximum Value:2015

3. Test Cases:

Input Data	Expected Outcome	Type
2,5,2000	Previous date: 1/5/2000	EP
1,3,2015	Previous date: 28/2/2015	EP
32,5,2000	Error	EP
15,13,2000	Error	EP
15,10,1899	Error	EP
1,1,1900	Error	BV
1,2,1900	Previous date:31/1/1900	BV
1,1,2015	Previous date: 31/12/2014	BV
31/12/2015	Previous date: 30/12/2015	BV

Executable code for the above is:

```
#include <iostream>
using namespace std;
bool isLeapYear(int year) {
if ((year % 400 == 0) || (year % 100 != 0 && year
% 4 == 0)) {
return true;
}
return false;
}
```

```
string previousDate(int day, int month, int year)
{
int daysInMonth[] = {31, 28, 31, 30, 31, 30, 31,
31, 30, 31, 30, 31};
if (isLeapYear(year)) {
daysInMonth[1] = 29;
}
if (year < 1900 || year > 2015 || month < 1 ||
month > 12 || day < 1 || day
> daysInMonth[month - 1]) {
return "Invalid Date";
}if (day == 1) {
if (month == 1) {
year--;
month = 12;
day = 31;
} else {
month--;
day = daysInMonth[month - 1];
}
} else {
day--;
}
return "Previous date is " + to
_
string(day) + "/" + to
_
string(month) + "/" +
to
_
string(year);
```

```
}
int main() {
cout << previousDate(32, 1, 2010) << endl;
cout << previousDate(0, 1, 2010) << endl;
cout << previousDate(1, 1, 1900) << endl;
cout << previousDate(15, 6, 2010) << endl;
cout << previousDate(1, 3, 2010) << endl;
cout << previousDate(1, 3, 2000) << endl;
cout << previousDate(1, 3, 1900) << endl;
cout << previousDate(29, 2, 2000) << endl;
cout << previousDate(30, 4, 2010) << endl;return
0;
}
```

Question-2

P1:

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

1. Equivalence Class Partitioning:

Wecan divide the inputs into valid and invalid equivalence classes.

- ❖ Valid Equivalence Classes:
  - v is present in the array a.
  - v is not present in the array a.
  - The array a contains one or more elements.

- ❖ Invalid Equivalence Classes:
  - The array a is empty.

2. Boundary Value Analysis:

Test boundary values for the size of the array a:

- Empty array (a with size 0).
- Array with one element.
- Array with two elements (minimum non-trivial size).
- Array with a large number of elements.

Boundary for the element v to be found:

- v is at the first index (index 0).
- v is at the last index (index a.length-1).

3.Test Cases

Input Data	**Expected Outcome**	**Type**

[1, 2, 3, 4, 5], 3	Return index: 2	EP
[10, 20, 30,40], 10	Return index: 10	EP
[9, 8, 7, 6],5	Error (Not Found)	EP
[], 5	Error (Array is empty)	EP
[3], 3	Return index: 0	BV
[1, 2], 2	Return index: 1	BV
[4, 5, 6], 6	Return index: 2	BV
[10, 20, 30, 40,..., 1000], 1000	Return index: 999	BV
[10, 20, 30, 40,..., 1000], 1	Error (Not Found)	BV

P2:

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

```
}
```

1. Equivalence Class Partitioning:

Wecan divide the inputs into valid and invalid equivalence classes.

- ❖ Valid Equivalence Classes:
  - v appears one or more times in the array a.
  - V does not appear in the array a.
  - The array a contains one or more elements.
- ❖ Invalid Equivalence Classes:
  - The Array a is empty.

2. Boundary Value Analysis:

Test boundary values for the size of the array a:

- Empty Array(a with size 0).
- Array With One Element.
- Array With Two Elements.
- Array With Large Number of elements.

Boundary For the element v Occurrence:

- v appears once.
- v appears multiple times.
- v does not appear at all.

Test Cases :

Input Data	Expected Outcome	Type

[1, 2, 3, 4, 5], 3	Return count: 1	EP
[10, 20, 30,40], 10	Return count: 1	EP
[9, 8, 7, 6],5	Error (Not Found)	EP
[], 5	Error (Array is empty)	EP
[3], 3	Return count: 1	BV
[1, 2], 2	Return count: 1	BV
[4, 5, 6], 6	Return count: 3	BV
[10, 20, 30, 40,..., 1000], 1000	Return count: 1	BV
[10, 20, 30, 40,..., 1000], 1	Error (Not Found)	BV

P3:

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

```
            return (mid);
        else if (v < a[mid])
             high= mid-1;
        Else
             low = mid+1;
     }
    return(-1);
 }
```

1. Equivalence Class Partitioning:

Wecan divide the inputs into valid and invalid equivalence classes.

- Valid Equivalence Classes:
  - v is present in the array a.
  - v is not present in the array a.
  - The array a contains one or more elements, and is sorted.

- Invalid Equivalence Classes:
  - The array a is empty.
  - Thearray a is unsorted.

2. Boundary Value Analysis:

Test boundary values for the size of the array a:

- Empty array (a with size 0).
- Array with one element.
- Array with two elements.

- Array with a large number of elements.

Boundary for the element v to be found:

- v is at the first index (index 0).
- v is at the last index (index a.length-1).
- v is not in the array at all.

Test Cases:

Input Data	Expected Outcome	Type
[1, 2, 3, 4, 5], 3	Return index: 2	EP
[10, 20, 30,40], 10	Return index: 10	EP
[9, 8, 7, 6],5	Error (Array is Unsorted)	EP
[5, 10, 15, 20], 25	Error (Not Found)	EP
[], 5	Error (Array is empty)	EP
[3], 3	Return index: 0	BV
[1, 2], 2	Return index: 1	BV
[4, 5, 6], 6	Return index: 2	BV
[10, 20, 30, 40,..., 1000], 1000	Return index: 999	BV
[10, 20, 30, 40,..., 1000], 1	Error (Not Found)	BV

P4:

```
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);
}
```

1. Equivalence Class Partitioning:

Wecan divide the inputs into valid and invalid equivalence classes.

- Valid Equivalence Classes:
  - Equilateral triangle (all three sides are equal).
  - Isosceles triangle (two sides are equal).
  - Scalene triangle (all sides are different).
- Invalid Equivalence Classes:

- Theside lengths do not satisfy the triangle inequality:
  - a >= b + c
  - b >= a + c
  - c >= a + b
- Oneormore sides are non-positive (i.e., a <= 0, b <= 0, c <=0).

2. Boundary Value Analysis:

Test boundary values for the lengths of the sides of the triangle:

- Minimal positive length (1).
- Equal side lengths for equilateral and isosceles.
- Slight variations in side lengths for scalene and invalid cases.
- Triangle inequality boundary conditions.

3. Test Cases:

Input Data	Expected Outcome	Type
(3, 3, 3)	Return: Equilateral(0)	EP
(5, 5, 9)	Return: Isosceles(1)	EP
(4, 5, 6)	Return: Scalene(2)	EP

(10, 5 ,4)	Error: (Triangular Inequality) (3)	EP
(0, 5, 5)	Error: (Non-positive Side Value) (3)	EP
(1, 1, 1)	Return: Equilateral(0)	BV
(2, 2, 3)	Return: Isosceles(1)	BV
(3, 4, 5)	Return: Scalene(2)	BV
(1, 2, 3)	Error: (Triangular Inequality) (3)	BV
(-2, 2, 3)	Error: (Non-positive Side Value) (3)	BV

P5:

```
public static boolean prefix(Strings1,Strings2)
 {
     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;
    }
```

1. Equivalence Class Partitioning:

Wecan divide the inputs into valid and invalid equivalence classes.

- Valid Equivalence Classes:
  - S1 is a valid prefix of s2.
  - S1 is not a prefix of s2.
  - S1 is an empty string (an empty string is a prefix of any string).
  - S1 is longer than s2.

2. Boundary Value Analysis:

Test boundary values for the lengths of s1 and s2:

- s1 and s2 are both empty strings.
- s1 is an empty string and s2 is non-empty.
- s1 has one character and s2 has the same character at the start.
- s1and s2 have the same length and are equal.
- s1 is longer than s2.

Test Cases:

Input Data	Expected Outcome	Type
(“pre, “prefix”)	Return: true	EP
(“fix”, “prefix”)	Return: false	EP
(“longer”, “short”)	Return: false	EP
(“prefix”, “prefix”)	Return: true	EP
(“a”, “abc”)	Return: true	BV
(“abc”, “abc”)	Return: true	Bv
(“abcdef”, “abc”)	Return: false	BV
(“abc”, “abx”)	Return: False	BV

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.

a) Identify the equivalence classes for the system:

Equivalence Classes:

Wecan identify different equivalence classes based on the properties of a triangle.

- **Valid Equivalence Classes:**
    - Equilateral Triangle: All sides are equal ($A = B = C$).
    - Isosceles Triangle: Two sides are equal ($A = B$ or $A = C$ or $B = C$).
    - Scalene Triangle: No sides are equal ($A \neq B \neq C$).
    - Right-Angled Triangle: The sides satisfy the Pythagorean theorem ($A^2 + B^2 = C^2$).

- **Invalid Equivalence Classes:**
    - Thesides do not satisfy the triangle inequality ($A + B \leq C$ or $A + C \leq B$orB $+ C \leq A$).
    - Oneormore sides are non-positive ($A \leq 0$ or $B \leq 0$ or $C \leq 0$).

b) Identify test cases to cover the identified equivalence classes.

Also, explicitly mention which test case would cover which

equivalence class.

Test Cases:

Input Data	**Expected Outcome**	**Equivalence Class**
(3.0, 3.0, 3.0)	Equilateral	Equilateral(A=B=C)
(6.0, 6.0, 8.0)	Isosceles	Isosceles(A=B,A≠C

		)
(3.0, 4.0, 6.0)	Scalene	Scalene(A≠B≠C)
(3.0, 4.0, 5.0)	Right Angled	Right Angled (A^2 + B^2 = C^2)
(1.0, 2.0, 3.0)	Error	Invalid
(0.0, 4.0, 5.0)	Error	Invalid
(-1.1, 2.0, 2.0)	Error	Invalid
(4.0, 4.0, 7.0)	Error	Invalid

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

This condition ensures that the sum of two sides is greater than the third.

Test Case 1:

- Input: (3.0, 4.0, 7.0) (Boundary value where A + B = C)
- Expected Outcome: Invalid Triangle (fails triangle inequality).

Test Case 2:

- Input: (4.0, 4.0, 7.0) (Boundary value where A + B = C)
- Expected Outcome: Isosceles Triangle.

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

This condition checks whether two sides of the triangle are equal.

Test Case:

- Input: (5.0, 7.0, 5.0) (Two sides are equal at the boundary).
- Expected Outcome: Isosceles Triangle.

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

This checks for cases where all three sides are equal.

Test Case 1:

- Input: (6.0, 6.0, 6.0) (All sides are equal at the boundary).
- Expected Outcome: Equilateral Triangle.

Test Case 2:

- Input: (7.0, 6.0, 6.0).
- Expected Outcome: Not an Equilateral Triangle.

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

This checks if the triangle satisfies the Pythagorean theorem.

Test Case:

- Input: (3.0, 4.0, 5.0) (Classic Pythagorean triplet).
- Expected Outcome: Right-Angled Triangle.

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

This tests the triangle inequality where the sum of two sides is not greater than the third side.

Test Case:

- Input: (1.0, 2.0, 3.0) (Boundary value where A + B = C).
- Expected Outcome: Invalid Triangle.

h) For non-positive input, identify test points.

This tests cases where one or more sides are non-positive.

Test Case 1:

- Input: (0.0, 3.0, 4.0) (Zero side length).
- Expected Outcome: Invalid Triangle.

Test Case 2:

- Input: (-1.0, 3.0, 3.0) (Negative side length).
- Expected Outcome: Invalid Triangle