IT314 Software Engineering

Lab 08

202201410 - Siddhant Kotak

Q1. Previous Date Determining Program

Equivalence Classes :

1. 1<=month<=12
2. month<1
3. month>12
4. 1<=day<=31
5. day<1
6. day>31
7. 1900<=year<=2015
8. year<1900
9. year>2015

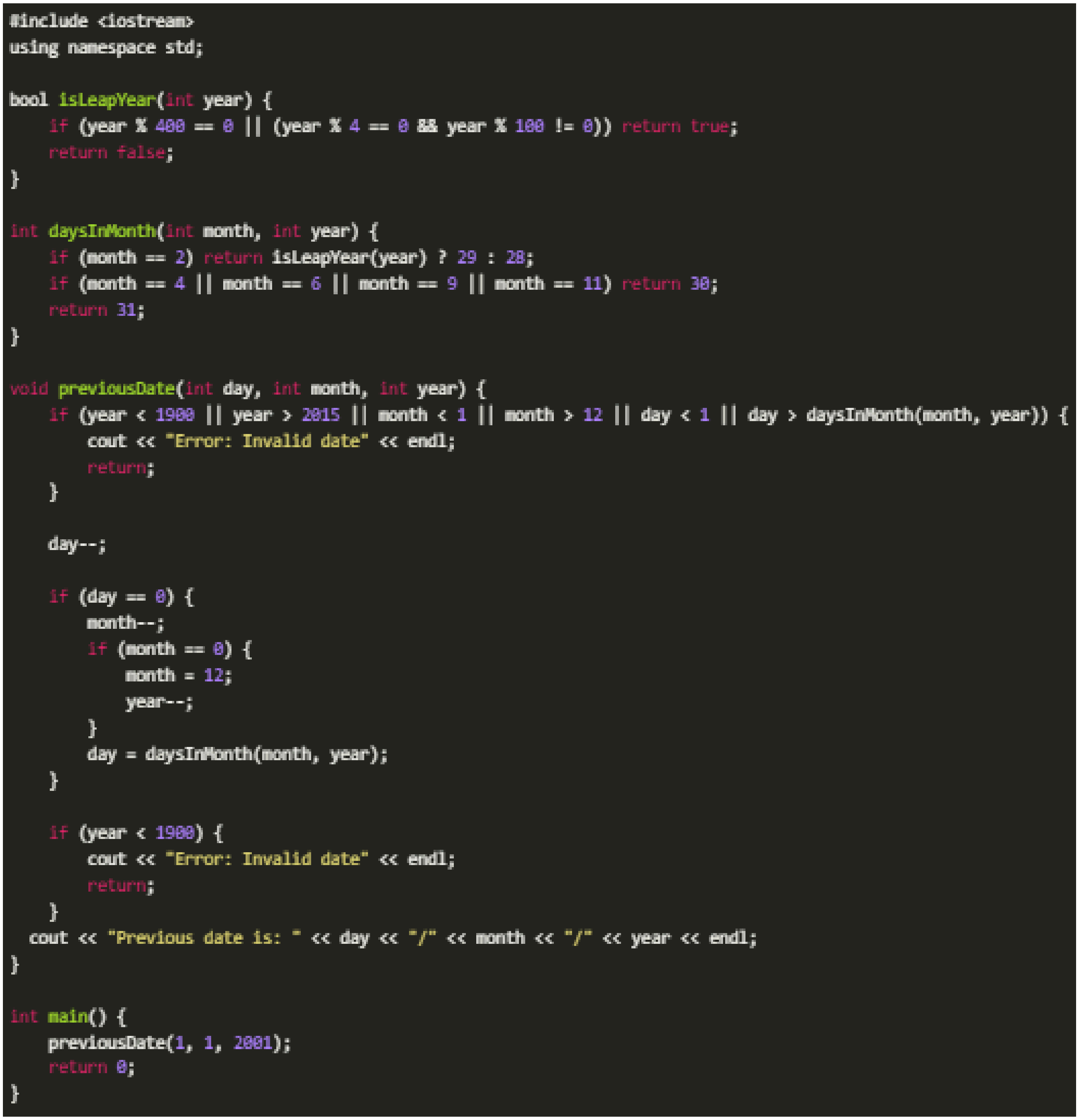
-- There are a total of **9** equivalence classes.

Boundary Value Analysis (Test Cases) :

Input (mm/dd/yyyy)	Expected Output	Equivalence Classes
10/31/2004	Valid Input (10/30/2004)	E1,E4,E7
0/1/1899	Invalid Input	E2,E4,E8
12/32/2015	Invalid Input	E1,E6,E7
13/0/2016	Invalid Input	E3,E5,E9
12/31/2015	Valid Input (12/30/2015)	E1,E4,E7
6/24/2009	Valid Input (6/23/2009)	E1,E4,E7

5/3/2001	Valid Input (5/2/2001)	E1,E4,E7
1/1/2000	Valid Input (31/12/1999)	E1,E4,E7

– Code:



```
#include <iostream>
using namespace std;

bool isLeapYear(int year) {
    if (year % 400 == 0 || (year % 4 == 0 && year % 100 != 0)) return true;
    return false;
}

int daysInMonth(int month, int year) {
    if (month == 2) return isLeapYear(year) ? 29 : 28;
    if (month == 4 || month == 6 || month == 9 || month == 11) return 30;
    return 31;
}

void previousDate(int day, int month, int year) {
    if (year < 1900 || year > 2015 || month < 1 || month > 12 || day < 1 || day > daysInMonth(month, year)) {
        cout << "Error: Invalid date" << endl;
        return;
    }

    day--;

    if (day == 0) {
        month--;
        if (month == 0) {
            month = 12;
            year--;
        }
        day = daysInMonth(month, year);
    }

    if (year < 1900) {
        cout << "Error: Invalid date" << endl;
        return;
    }
  cout << "Previous date is: " << day << "/" << month << "/" << year << endl;
}

int main() {
    previousDate(1, 1, 2001);
    return 0;
}
```

-- P1 : Linear Search Program

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

Equivalence Classes :

1. The value **v** occurs once in the array **a**.
2. The value **v** occurs multiple times in array **a**.
3. The value **v** is not present in the array **a**.
4. The value **v** is at the first or at the last position of the array **a.**
5. The array **a** is empty.

Boundary Value Analysis (Test Cases) :

Input (v,a[])	Expected Output	Equivalence Classes
1,[3,2,3,1,5]	3	E1
5,[]	-1	E5
2,[3,2,2,4,2,5]	1	E2
2,[2,4,5]	0	E4
3,[4,5,6]	-1	E3
4,[1,2,4]	2	E4

-- P2 : Count Item Program

```
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 Classes :

1. The value **v** occurs once in the array **a**.
2. The value **v** occurs multiple times in array **a**.
3. The value **v** is not present in the array **a**.
4. The value **v** is at the first or at the last position of the array **a.**
5. The array **a** is empty.

Boundary Value Analysis (Test Cases) :

Input (v,a[])	Expected Output	Equivalence Classes
1,[3,2,3,1,5]	1	E1
5,[]	0	E5
2,[3,2,2,4,2,5]	3	E2
2,[2,4,5]	1	E4
3,[4,5,6]	0	E3
4,[1,2,4]	1	E4

-- P3 : Binary Search Program

```
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);
}
```

Equivalence Classes :

1. The value **v** occurs once in the array **a**.
2. The value **v** occurs multiple times in array **a**.
3. The value **v** is smaller than the first element in the array **a**.
4. The value **v** is larger than the last element in the array **a**.
5. The value **v** is at the first or at the last position of the array **a.**
6. The array **a** is empty.

Boundary Value Analysis (Test Cases) :

Input (v,a[])	Expected Output	Equivalence Classes
1,[1,2,3,3,5]	0	E1,E5
5,[]	-1	E6

3,[2,3,3,3,4,5]	1	E2
1,[2,4,5]	-1	E3
9,[4,5,6]	-1	E4
4,[1,2,4]	2	E5

-- P5 : Prefix of string Program

```
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;
}
```

Equivalence Classes :

1. **s1** is a non-empty string and is a prefix of **s2.**
2. **s1** is an empty string, which is considered a prefix of any string **s2**.
3. **s1** is a non-empty string, but **s2** is empty.
4. **s1** is equal to **s2**.
5. **s1** is longer than **s2**.
6. **s1** is not a prefix of **s2** (they differ after some characters).

Boundary Value Analysis (Test Cases) :

Input (s1,s2)	Expected Output	Equivalence Classes
“soft”,“software”	YES	E1
“”,“software”	YES	E2
“soft”,“”	NO	E3
“soft”,“hardware”	NO	E6
“software”,“soft”	NO	E5
“software”,“software”	YES	E4

-- P6 : Triangle Classification Program with floating input values

(a) Identifying Equivalence Classes

Valid Triangles:

- Equilateral: All sides are equal (A = B = C).
- Isosceles: Two sides are equal (A = B, A = C, or B = C).
- Scalene: No sides are equal, and they satisfy the triangle inequality (A + B > C, A + C > B, B + C > A).
- Right-angled: $A^2 + B^2 = C^2$.

Invalid Triangles:

- Non-Triangle: The sides do not satisfy the triangle inequality (A + B ≤ C, A + C ≤ B, B + C ≤ A).
- Non-positive Values: One or more sides are zero or negative (A ≤ 0, B ≤ 0, C ≤ 0).

(b) Test Cases to cover Equivalence Classes

Input (A,B,C)	Expected Output	Equivalence Class
3.0,3.0,3.0	Equilateral	Equilateral (A = B = C)
3.0,4.0,5.0	Right-angled	Scalene, Right-angled
3.0,5.0,5.0	Isosceles	Isosceles (A = B)
1.0,2.0,3.0	Invalid Triangle	Non-triangle (A + B = C)
0.0,4.0,5.0	Invalid Triangle	Non-positive input (A ≤ 0)
2.0,3.0,4.0	Scalene	Scalene (A + B > C)
-3.0,4.0,5.0	Invalid Triangle	Non-positive input (A < 0)

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

Input (A,B,C)	Expected Output	Boundary Condition
3.0,3.0,5.9	Scalene	A + B > C
3.0,3.0,6.0	Invalid Triangle	A + B = C

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

Input (A,B,C)	Expected Output	Boundary Condition
5.0,3.0,5.0	Isosceles	A = C
5.0,3.1,5.0	Scalene	A not equal to C

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

Input (A,B,C)	Expected Output	Boundary Condition
4.0,4.0,4.0	Equilateral	A = B = C
4.0,4.1,4.0	Isosceles	A = C but != B

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

Input (A,B,C)	Expected Output	Boundary Condition
3.0,4.0,5.0	Right-angled	$A^2 + B^2 = C^2$
3.0,4.0,5.1	Scalene	$A^2 + B^2 != C^2$

g) Non-Triangle Case (A + B ≤ C)

Input (A,B,C)	Expected Output	Boundary Condition
1.0,2.0,3.0	Invalid Triangle	A + B = C
1.0,1.0,3.0	Invalid Triangle	A + B < C

h) Non-Positive Input

Input (A,B,C)	Expected Output	Equivalence Class
0.0,4.0,5.0	Invalid Triangle	A = 0
-3.0,4.0,5.0	Invalid Triangle	A < 0
3.0,0.0,5.0	Invalid Triangle	B = 0