Dhirubhai Ambani
Institute of Information and Communication Technology

Software Engineering Lab - 08

Name - Meet Chauhan
Id - 202201262

Q1.
Ans:-

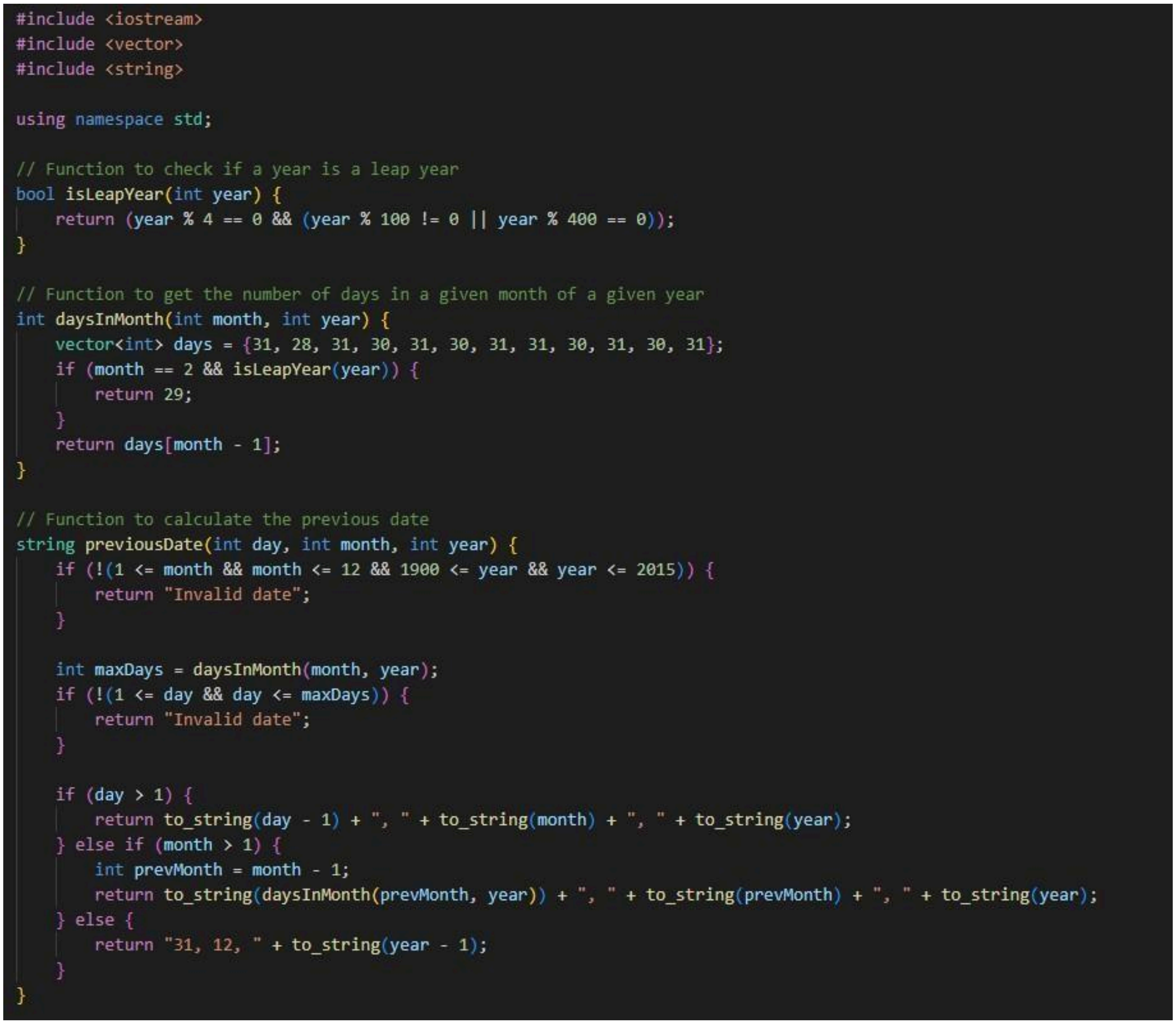
Equivalence classes for the given question are -

1) month <1, 1<= day <= 31 , 1900 <= year <= 2015
2) month <1, 1<= day <= 31, year < 1900
3) month <1, 1<= day <= 31, year > 2015
4) month<1, day<1, 1900 <= year <= 2015
5) month <1, day<1, year<1900
6) month <1, day<1, year > 2015
7) month<1, day>31, 1900 <= year <= 2015
8) month <1, day>31, year<1900
9) month <1, day>31, year > 2015
10) 1 <= month <= 12 , 1<= day <= 31 , 1900 <= year <= 2015
11) 1 <= month <= 12 , 1<= day <= 31 , year < 1900
12) 1 <= month <= 12 , 1<= day <= 31 , year > 2015
13) 1 <= month <= 12 , day<1, 1900 <= year <= 2015
14) 1 <= month <= 12 , day<1, year<1900
15) 1 <= month <= 12 , day<1, year >2015
16) 1 <= month <= 12, day>31, 1900 <= year <= 2015
17) 1 <= month <= 12, day>31, year<1900
18) 1 <= month <= 12, day>31, year > 2015
19) month >12 , 1<= day <= 31 , 1900 <= year <= 2015
20) month >12 , 1<= day <= 31 , year < 1900
21) month >12, 1<= day <= 31 , year > 2015
22) month >12 , day<1, 1900 <= year <= 2015
23) month >12 , day<1, year<1900
24) month >12 , day<1, year >2015
25) month >12, day>31, 1900 <= year <= 2015
26) month >12, day>31, year<1900
27) month >12, day>31, year > 2015

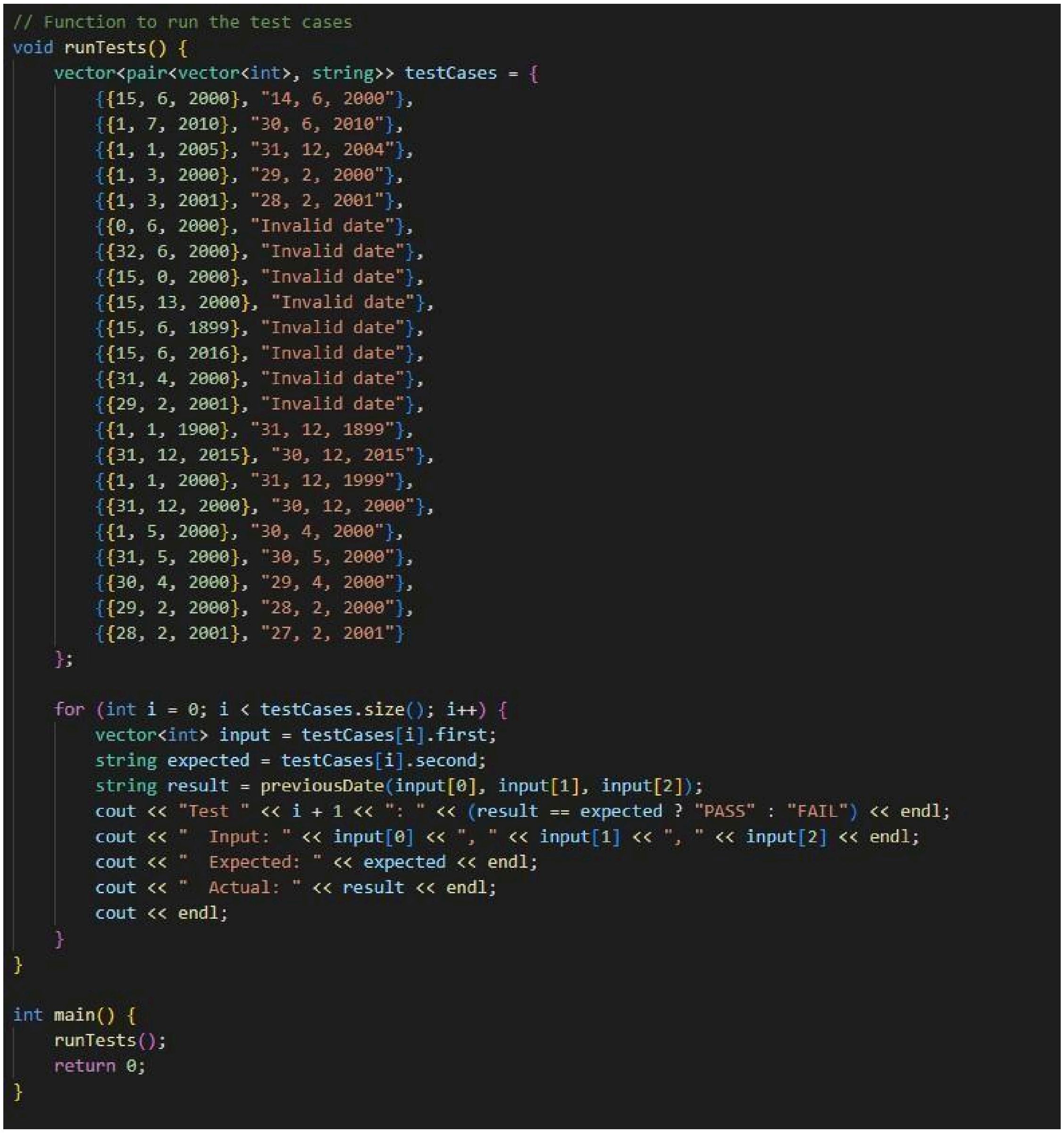
Set of test cases: -

1) 0,20,2001 - Invalid date
2) 0, 21,1801 - Invalid date
3) 0, 22,2018 - Invalid date
4) 0, 0,2002 - Invalid date
5) 0, 0,1802 - Invalid date
6) 0, 0,2019 - Invalid date
7) 0, 32,2003 - Invalid date
8) 0, 33,1803 - Invalid date
9) 0, 34,2020 - Invalid date
10) 1,3,2004 - valid date
11) 2,1,1804 - previous date
12) 3,2,2021 - Future date
13) 4,0,2005 - Invalid date
14) 5,0,1805 - Invalid date
15) 6,0,2022 - Invalid date
16) 7,35,2006 - Invalid date
17) 8,36,1806 - Invalid date
18) 9,37,2023 - Invalid date
19) 13,4,2007 - Invalid date
20) 14,5,1807 - Invalid date
21) 15,6,2024 - Invalid date
22) 16,0,2008 - Invalid date
23) 17,0,1808 - Invalid date
24) 18,0,2025 - Invalid date
25) 19,41,2009 - Invalid date
26) 20,42,1809 - Invalid date
27) 21,43,2026 - Invalid date

c++ implementation :



```cpp
#include <iostream>
#include <vector>
#include <string>

using namespace std;

// Function to check if a year is a leap year
bool isLeapYear(int year) {
    return (year % 4 == 0 && (year % 100 != 0 || year % 400 == 0));
}

// Function to get the number of days in a given month of a given year
int daysInMonth(int month, int year) {
    vector<int> days = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
    if (month == 2 && isLeapYear(year)) {
        return 29;
    }
    return days[month - 1];
}

// Function to calculate the previous date
string previousDate(int day, int month, int year) {
    if (!(1 <= month && month <= 12 && 1900 <= year && year <= 2015)) {
        return "Invalid date";
    }

    int maxDays = daysInMonth(month, year);
    if (!(1 <= day && day <= maxDays)) {
        return "Invalid date";
    }

    if (day > 1) {
        return to_string(day - 1) + ", " + to_string(month) + ", " + to_string(year);
    } else if (month > 1) {
        int prevMonth = month - 1;
        return to_string(daysInMonth(prevMonth, year)) + ", " + to_string(prevMonth) + ", " + to_string(year);
    } else {
        return "31, 12, " + to_string(year - 1);
    }
}
```



```cpp
// Function to run the test cases
void runTests() {
    vector<pair<vector<int>, string>> testCases = {
        {{15, 6, 2000}, "14, 6, 2000"},
        {{1, 7, 2010}, "30, 6, 2010"},
        {{1, 1, 2005}, "31, 12, 2004"},
        {{1, 3, 2000}, "29, 2, 2000"},
        {{1, 3, 2001}, "28, 2, 2001"},
        {{0, 6, 2000}, "Invalid date"},
        {{32, 6, 2000}, "Invalid date"},
        {{15, 0, 2000}, "Invalid date"},
        {{15, 13, 2000}, "Invalid date"},
        {{15, 6, 1899}, "Invalid date"},
        {{15, 6, 2016}, "Invalid date"},
        {{31, 4, 2000}, "Invalid date"},
        {{29, 2, 2001}, "Invalid date"},
        {{1, 1, 1900}, "31, 12, 1899"},
        {{31, 12, 2015}, "30, 12, 2015"},
        {{1, 1, 2000}, "31, 12, 1999"},
        {{31, 12, 2000}, "30, 12, 2000"},
        {{1, 5, 2000}, "30, 4, 2000"},
        {{31, 5, 2000}, "30, 5, 2000"},
        {{30, 4, 2000}, "29, 4, 2000"},
        {{29, 2, 2000}, "28, 2, 2000"},
        {{28, 2, 2001}, "27, 2, 2001"}
    };

    for (int i = 0; i < testCases.size(); i++) {
        vector<int> input = testCases[i].first;
        string expected = testCases[i].second;
        string result = previousDate(input[0], input[1], input[2]);
        cout << "Test " << i + 1 << ": " << (result == expected ? "PASS" : "FAIL") << endl;
        cout << "  Input: " << input[0] << ", " << input[1] << ", " << input[2] << endl;
        cout << "  Expected: " << expected << endl;
        cout << "  Actual: " << result << endl;
        cout << endl;
    }
}

int main() {
    runTests();
    return 0;
}
```

Q2.

Problem 1:

Equivalence Partitioning

Input Data	Expected Outcome
5, {1, 2, 3}	-1
2, {1, 2, 3}	1
-1, {-1, 0, 1}	0
1, {}	-1
4, {4}	0
1, {1, 2, 3}	0
3, {1, 2, 3}	2
null, {1, 2, 3}	An Error message
{1, 2, 3}, null	An Error message

Boundary Value Analysis:

Input Data	Expected Outcome
5, {}	-1
-2147483648, {-2147483648, 0, 2147483647}	0
2147483647, {-2147483648, 0, 2147483647}	2
1, {1, 2}	0
2, {1, 2}	1
4, {1, 2, 3}	-1
5, null	An Error message
{1, 2, 3}, {}	An Error message

Problem 2 :

Equivalence Partitioning:

Input Data	**Expected Outcome**
5, {1, 2, 3}	0
2, {1, 2, 3}	1
-1, {-1, 0, 1}	1
1, {}	0
4, {4, 4, 4}	3
1, {1, 2, 3, 1, 1}	3
3, {1, 2, 3, 3, 3, 3}	4
null, {1, 2, 3}	An Error message
{1, 2, 3}, null	An Error message

Boundary Value Analysis:

Input Data	**Expected Outcome**
5, {}	0
-2147483648, {-2147483648, 0, 2147483647}	1
2147483647, {-2147483648, 0, 2147483647}	1
1, {1, 2}	1
2, {1, 2, 2}	2
4, {1, 2, 3}	0
5, null	An Error message
{1, 2, 3}, {}	An Error message

Problem 3 :

Equivalence Partitioning:

Input Data	Expected Outcome
5, {1, 2, 3}	-1
2, {1, 2, 3}	1
1, {1, 2, 3}	0
3, {1, 2, 3}	2
4, {1, 4, 6, 8}	1
0, {0, 1, 2, 3}	0
100, {10, 20, 30, 100}	3
null, {1, 2, 3}	An Error message
{1, 2, 3}, null	An Error message

Boundary Value Analysis:

Input Data	Expected Outcome
5, {}	-1
-2147483648, {-2147483648, 0, 2147483647}	0
2147483647, {-2147483648, 0, 2147483647}	2
1, {1, 2}	0
2, {1, 2}	1
4, {1, 2, 3}	-1
5, null	An Error message
{1, 2, 3}, {}	An Error message

Problem 4 :

Equivalence Partitioning:

Input Data	Expected Outcome
3, 3, 3	EQUILATERAL (0)
3, 3, 2	ISOSCELES (1)
3, 4, 5	SCALENE (2)
1, 2, 3	INVALID (3)
1, 1, 2	INVALID (3)
5, 1, 1	INVALID (3)
2, 2, 3	ISOSCELES (1)
0, 1, 1	An Error message
1, 0, 1	An Error message

Boundary Value Analysis:

Input Data	Expected Outcome
1, 1, 1	EQUILATERAL (0)
1, 1, 2	INVALID (3)
2, 2, 4	INVALID (3)
2, 3, 5	INVALID (3)
3, 4, 7	INVALID (3)
1, 2, 2	ISOSCELES (1)
1, 2, 3	INVALID (3)
0, 1, 1	An Error message
1, 1, 0	An Error message

Problem 5:

Equivalence Partitioning:

Input Data	Expected Outcome
"pre", "prefix"	true
"pre", "postfix"	false
"prefix", "pre"	false
"test", "test"	true
"", "anything"	true
"anything", ""	false
"pre", "preparation"	true
null, "prefix"	An Error message
"prefix", null	An Error message

Boundary Value Analysis:

Input Data	Expected Outcome
"test", ""	false
"a", "a"	true
"a", "b"	false
"", ""	true
"start", "startmiddle"	true
"longprefix", "short"	false
"short", "longprefix"	true
null, "anything"	An Error message
"anything", null	An Error message

Problem 6:

a) Identify the Equivalence Classes

Equilateral Triangle: All three sides are equal.

Isosceles Triangle: Exactly two sides are equal.

Scalene Triangle: No sides are equal.

Right-Angled Triangle: Satisfies a2+b2=c2.

Invalid Triangle: Does not satisfy the triangle inequality a+b>c.

Non-positive Input: One or more sides are non-positive.

b) Identify Test Cases to Cover the Equivalence Classes

Equivalence Partitioning:

Input Data	Expected Outcome	Equivalence Class
3.0, 3.0, 3.0	Equilateral	Equilateral Triangle
3.0, 3.0, 2.0	Isosceles	Isosceles Triangle
3.0, 4.0, 5.0	Scalene	Scalene Triangle
3.0, 4.0, 0.0	Invalid	Invalid Triangle
0.0, 0.0, 0.0	Invalid	Non-positive Input
5.0, 1.0, 1.0	Invalid	Invalid Triangle
3.0, 4.0, 6.0	Scalene	Scalene Triangle

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

Boundary Value Analysis:

Input Data	Expected Outcome
2.0, 2.0, 3.99	Scalene
2.0, 2.0, 4.0	Invalid
2.0, 2.0, 4.01	Invalid

d) Boundary Condition A = C (Isosceles Triangle)

Boundary Value Analysis:

Input Data	Expected Outcome
3.0, 4.0, 3.0	Isosceles
3.0, 3.0, 3.0	Equilateral
3.0, 3.0, 4.0	Isosceles

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

Boundary Value Analysis:

Input Data	Expected Outcome
3.0, 3.0, 3.0	Equilateral
1.0, 1.0, 1.0	Equilateral
2.5, 2.5, 2.5	Equilateral

f) Boundary Condition A2+B2=C2 (Right-Angle Triangle)

Boundary Value Analysis:

Input Data	Expected Outcome
3.0, 4.0, 5.0	Right Angled
6.0, 8.0, 10.0	Right Angled
5.0, 12.0, 13.0	Right Angled

g) Non-Triangle Case

Boundary Value Analysis:

Input Data	Expected Outcome
1.0, 2.0, 3.0	Invalid
1.0, 2.0, 4.0	Invalid
1.0, 1.0, 2.0	Invalid

h) Non-Positive Input

Boundary Value Analysis:

Input Data	**Expected Outcome**
0.0, 1.0, 1.0	Invalid
-1.0, 1.0, 1.0	Invalid
1.0, 0.0, 1.0	Invalid