Dhirubhai Ambani
Institute of Information and Communication Technology

Lab08: Functional Testing (Black-Box)

IT314 –Software Engineering
Jyot Vasava
202201169

ProgramSpecificationandTestCases

1. ProgramSpecification

Input:Tripleofday,month,andyear

Inputranges:
1<=month<=12
1<=day<=31
1900<=year<=2015

Output:Previousdateor"Invaliddate"

2. TestSuite

2.1 EquivalencePartitioning

ValidPartitions:

- Normaldays(notmonthendoryearend)
- Monthend(notyearend)
- Yearend(December31)
- LeapyearFebruary29

InvalidPartitions:

- Invalidmonth(<1or>12)
- Invalidday(<1or>maxdaysinmonth)
- Invalidyear(<1900or>2015)
- Invaliddayforspecificmonth(e.g.,February30)

2.2 BoundaryValueAnalysis

- Firstdayofyear:January1,YYYY
- Lastdayofyear:December31,YYYY

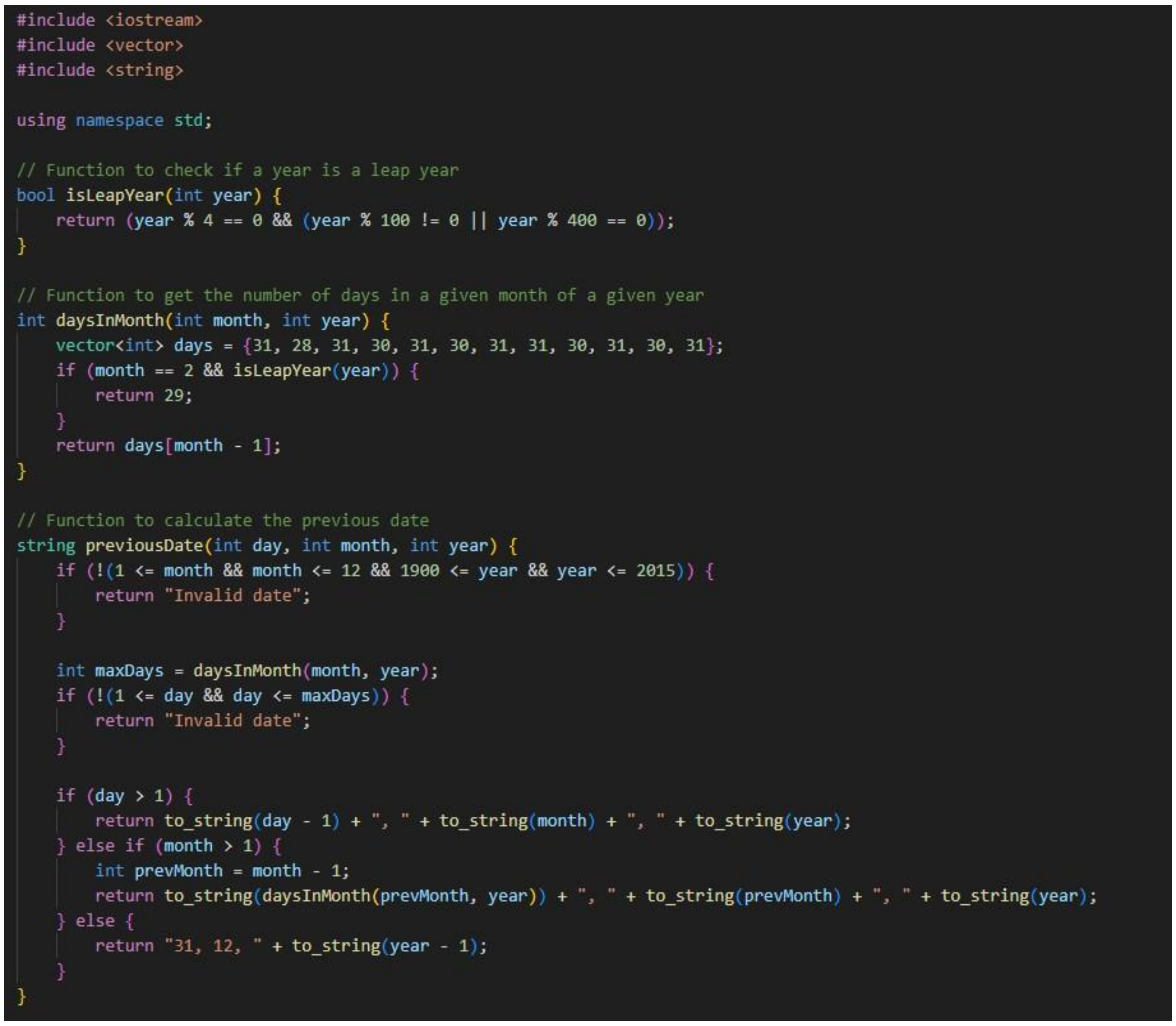
- Firstdayofmonth:DD1,MM
- Lastdayofmonth:DD30/31,MM(28/29forFebruary)
- Minimumvalidyear:1900
- Maximumvalidyear:2015

2.3 TestCases

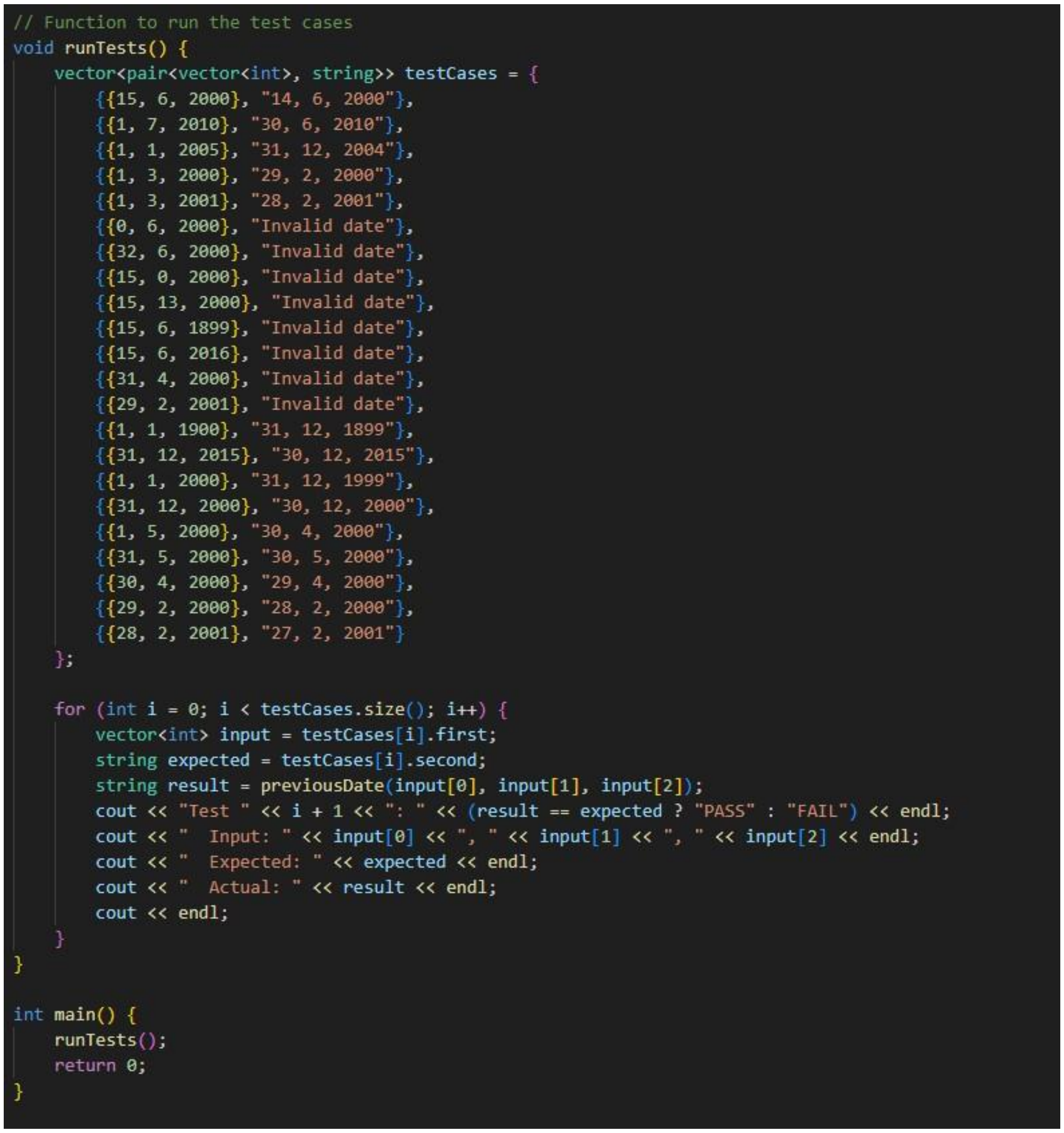
Tester Action andInput Data	**ExpectedOutcome**	**Remarks**
a,b,c	AnErrormessage	Invalidinputformat
15,6,2000	14,6,2000	Normalday
1,7,2010	30,6,2010	Monthend
1,1,2005	31,12,2004	Yearend
1,3,2000	29,2,2000	Leapyear
1,3,2001	28,2,2001	Non-leapyear
0,6,2000	Invaliddate	Invalidday(toolow)
32,6,2000	Invaliddate	Invalidday(toohigh)
15,0,2000	Invaliddate	Invalidmonth(toolow)
15,13,2000	Invaliddate	Invalidmonth(toohigh)
15,6,1899	Invaliddate	Invalidyear(toolow)
15,6,2016	Invaliddate	Invalidyear(toohigh)
31,4,2000	Invaliddate	InvaliddayforApril
29,2,2001	Invaliddate	InvaliddayforFebruaryinn on-leapyear
1,1,1900	31,12,1899	Boundary:Minimumvalidy ear-1

31,12,2015	30,12,2015	Boundary:Maximumvalid year
1,1,2000	31,12,1999	Boundary:Firstdayofyea r
31,12,2000	30,12,2000	Boundary:Lastdayofyea r
1,5,2000	30,4,2000	Boundary:Firstdayofmo nth
31,5,2000	30,5,2000	Boundary:Lastdayof31-daymonth
30,4,2000	29,4,2000	Boundary:Lastdayof30-daymonth
29,2,2000	28,2,2000	Boundary:LastdayofFe bruaryinleapyear
28,2,2001	27,2,2001	Boundary:LastdayofFeb ruaryinnon-leapyear

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;
}
```

Problem1:

EquivalencePartitioning

InputData	ExpectedOutcome
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}	AnErrormessage
{1,2,3},null	AnErrormessage

BoundaryValueAnalysis:

InputData	ExpectedOutcome
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	AnErrormessage
{1,2,3},{}	AnErrormessage

Problem2:

EquivalencePartitioning:

InputData	ExpectedOutcome
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}	AnErrormessage
{1,2,3},null	AnErrormessage

BoundaryValueAnalysis:

InputData	ExpectedOutcome
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 Errormessage
{1,2,3},{}	An Errormessage

Problem3:

EquivalencePartitioning:

InputData	ExpectedOutcome
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}	AnErrormessage
{1,2,3},null	AnErrormessage

BoundaryValueAnalysis:

InputData	ExpectedOutcome
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 Errormessage
{1,2,3},{}	An Errormessage

Problem4:

EquivalencePartitioning:

InputData	ExpectedOutcome
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	AnError message
1,0,1	AnError message

BoundaryValueAnalysis:

InputData	ExpectedOutcome
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	AnError message
1,1,0	AnError message

Problem5:

EquivalencePartitioning:

InputData	ExpectedOutcome
"pre","prefix"	true
"pre","postfix"	false
"prefix","pre"	false
"test","test"	true
"","anything"	true
"anything",""	false
"pre","preparation"	true
null,"prefix"	AnError message
"prefix",null	AnError message

BoundaryValueAnalysis:

InputData	ExpectedOutcome
"test", ""	false
"a", "a"	true
"a", "b"	false
"",""	true
"start","startmiddle"	true
"longprefix","short"	false
"short","longprefix"	true
null,"anything"	AnErrormessage
"anything",null	AnErrormessage

Problem6:

a)IdentifytheEquivalenceClasses

Equilateral Triangle: All three sides are

equal.Isosceles Triangle: Exactly two sides are

equal.ScaleneTriangle:Nosidesareequal.

Right-AngledTriangle:Satisfiesa2+b2=c2.

Invalid Triangle: Does not satisfy the triangle inequality

a+b>c.Non-positive Input:Oneor moresidesarenon-positive.

b) IdentifyTestCases toCovertheEquivalenceClasses

EquivalencePartitioning:

InputData	**ExpectedOutcome**	**EquivalenceClass**
3.0,3.0,3.0	Equilateral	EquilateralTriangle
3.0,3.0,2.0	Isosceles	IsoscelesTriangle
3.0,4.0,5.0	Scalene	ScaleneTriangle
3.0,4.0,0.0	Invalid	InvalidTriangle
0.0,0.0,0.0	Invalid	Non-positiveInput
5.0,1.0,1.0	Invalid	InvalidTriangle
3.0,4.0,6.0	Scalene	ScaleneTriangle

c) BoundaryCondition A+ B>C(ScaleneTriangle)

BoundaryValueAnalysis:

InputData	**ExpectedOutcome**
2.0,2.0,3.99	Scalene
2.0,2.0,4.0	Invalid
2.0,2.0,4.01	Invalid

d)BoundaryConditionA=C(IsoscelesTriangle)

BoundaryValueAnalysis:

InputData	ExpectedOutcome
3.0,4.0,3.0	Isosceles
3.0,3.0,3.0	Equilateral
3.0,3.0,4.0	Isosceles

e) BoundaryCondition A=B=C(EquilateralTriangle)

BoundaryValueAnalysis:

InputData	ExpectedOutcome
3.0,3.0,3.0	Equilateral
1.0,1.0,1.0	Equilateral
2.5,2.5,2.5	Equilateral

f) BoundaryConditionA2+B2=C2(Right-AngleTriangle)

BoundaryValueAnalysis:

InputData	ExpectedOutcome
3.0,4.0,5.0	RightAngled
6.0,8.0,10.0	RightAngled
5.0,12.0,13.0	RightAngled

g) Non-TriangleCase

BoundaryValueAnalysis:

InputData	ExpectedOutcome
1.0,2.0,3.0	Invalid
1.0,2.0,4.0	Invalid
1.0,1.0,2.0	Invalid

h)Non-PositiveInput

BoundaryValueAnalysis:

InputData	**ExpectedOutcome**
0.0,1.0,1.0	Invalid
-1.0,1.0, 1.0	Invalid
1.0,0.0,1.0	Invalid