Rabadiya Utsav

202201081

LAB 8

Question 1: Test Case Design for Date Program

Input Domains:

- Day: 1 ≤ day ≤ 31
- Month: 1 ≤ month ≤ 12
- Year: 1900 ≤ year ≤ 2015

1. Equivalence Partitioning Test Cases

Valid Equivalence Classes:

1. Regular days (not month end): 2-27
2. Month end days: 28,30,31 (depending on month)
3. Regular months: 1-12
4. Regular years: 1900-2015
5. Leap years: divisible by 4 (except century years not divisible by 400)

Invalid Equivalence Classes:

1. Invalid days: < 1 or > 31
2. Invalid months: < 1 or > 12
3. Invalid years: < 1900 or > 2015

Test Case ID	**Input (dd,mm,yyyy)**	**Expected Output**	**Class Covered**
1	15,06,2000	14,06,2000	Regular day
2	01,03,2000	29,02,2000	Month end (leap year)
3	01,03,2001	28,02,2001	Month end (non-leap year)
4	01,05,2000	30,04,2000	Month with 30 days

Test Case ID	Input (dd,mm,yyyy)	Expected Output	Class Covered
5	32,05,2000	Invalid date	Invalid day
6	15,13,2000	Invalid date	Invalid month
7	15,06,2016	Invalid date	Invalid year

2. Boundary Value Analysis Test Cases

Test Case ID	Input (dd,mm,yyyy)	Expected Output	Boundary Type
1	01,01,1900	31,12,1899	Invalid (year < 1900)
2	01,01,2015	31,12,2014	Valid year boundary
3	01,01,1900	31,12,1899	Lower year boundary
4	31,12,2015	30,12,2015	Upper year boundary
5	01,01,2000	31,12,1999	First day of month
6	31,01,2000	30,01,2000	Last day of month

Code:

```
#include <iostream>

#include <string>

using namespace std;
bool isLeapYear(int year) {
return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
}
string previousDate(int day, int month, int year) {
if (year < 1900 || year > 2015 || month < 1 || month > 12 || day < 1
|| day > 31) {
return "Error: Invalid date";
}
int daysInMonth[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
if (isLeapYear(year)) daysInMonth[1] = 29;
if (day > daysInMonth[month - 1]) return "Error: Invalid date";
if (day > 1) day--;
else {
month--;
if (month < 1) {
month = 12;
year--;
}
day = daysInMonth[month - 1];
}
return to_string(day) + "/" + to_string(month) + "/" +
to_string(year);
}
int main() {
cout << previousDate(15, 7, 2000) << endl; // Expected: 14/7/2000
cout << previousDate(1, 1, 1900) << endl; // Expected: 31/12/1899
cout << previousDate(32, 7, 2000) << endl; // Expected: Error:
Invalid date
return 0;
}
```

Question 2: Analysis of Given Programs

P1. Linear Search Analysis

Test Case ID	Input Array	Search Value	Expected Output
1	[1,2,3,4,5]	3	2
2	[1,2,3,4,5]	6	-1
3	[]	1	-1
4	[1,1,1]	1	0
5	[1]	1	0

P2. Count Item Analysis

Test Case ID	Input Array	Search Value	Expected Output
1	[1,2,3,2,4]	2	2
2	[1,2,3,4,5]	6	0
3	[]	1	0
4	[1,1,1]	1	3
5	[1]	1	1

P3. Binary Search Analysis

Test Case ID	Input Array	Search Value	Expected Output
1	[1,2,3,4,5]	3	2
2	[1,2,3,4,5]	6	-1
3	[]	1	-1
4	[1,1,1]	1	0,1,or 2
5	[1]	1	0

P4. Triangle Classification Analysis

Test Case ID	Input (a,b,c)	Expected Output	Description
1	3,3,3	EQUILATERAL	Equal sides
2	3,3,4	ISOSCELES	Two equal sides
3	3,4,5	SCALENE	No equal sides
4	1,1,3	INVALID	a+b <= c
5	0,1,1	INVALID	Zero side

P5. String Prefix Analysis

Test Case ID	s1	s2	Expected Output	Description
1	"hello"	"hello world"	true	Valid prefix
2	"world"	"hello"	false	Longer than string
3	""	"hello"	true	Empty prefix
4	"hey"	"hello"	false	Different chars
5	"hello"	"hello"	true	Equal strings

P6: Triangle Classification (Floating Point)

Equivalence Classes:

1. Valid triangles:

    - Scalene (all sides different)
    - Isosceles (two sides equal)
    - Equilateral (all sides equal)
    - Right-angled ($a^2 + b^2 = c^2$)

2. Invalid triangles:

    - Sum of two sides ≤ third side
    - Negative or zero sides
    - Non-numeric input

Test Cases:

a) Equivalence Classes Coverage:

Test Case ID	Input (A,B,C)	Expected Output	Class
1	3.0,4.0,5.0	Scalene	Valid scalene
2	5.0,5.0,6.0	Isosceles	Valid isosceles
3	6.0,6.0,6.0	Equilateral	Valid equilateral
4	1.0,1.0,3.0	Invalid	Invalid triangle
5	-1.0,2.0,2.0	Invalid	Negative input

b) Boundary Tests:

For A + B > C (Scalene):

1: (2.0, 2.0, 3.99)
2: (2.0, 2.0, 4.0)

For A = C (Isosceles):

3: (5.0, 4.0, 5.0)
4: (5.0, 5.1, 5.0)

For A = B = C (Equilateral):

5: (5.0, 5.0, 5.0)
6: (5.0, 5.0, 5.1)

For Right-angle ($A^2 + B^2 = C^2$):

7: (3.0, 4.0, 5.0)
8: (5.0, 12.0, 13.0)

For Non-triangle:

9: (1.0, 1.0, 2.0)
10: (1.0, 1.0, 2.1)

For Non-positive input:

11: (0.0, 1.0, 1.0)
12: (-1.0, 1.0, 1.0)