IT314 : Software Engineering

Name : Devansh Modi
ID : 202201198

Question 1 :

❖ Equivalence Partitioning Test Cases

Valid Date Equivalence Classes:-

1. **Valid Date (Regular Day)**:

   Input: (15, 7, 2012)

   Expected Outcome: (14, 7, 2012)

2. **Valid Date (End of February, Leap Year)**:

   Input: (29, 2, 2000)

   Expected Outcome: (28, 2, 2000)

3. **Valid Date (End of February, Non-Leap Year)**:

   Input: (28, 2, 2015)

   Expected Outcome: (27, 2, 2015)

4. **Valid Date (End of Month)**:

   Input: (30, 4, 2011)

   Expected Outcome: (29, 4, 2011)

5. **Valid Date (Month with 30 Days)**:

   Input: (30, 9, 2017)

   Expected Outcome: (29, 9, 2017)

Invalid Date Equivalence Classes:-

1. **Invalid Day (Zero Day)**:

   Input: (0, 12, 2014)

   Expected Outcome: An Error message
2. **Invalid Day (Negative Day)**:

   Input: (-1, 11, 2013)

   Expected Outcome: An Error message
3. **Invalid Month (Zero Month)**:

   Input: (1, 0, 2010)

   Expected Outcome: An Error message
4. **Invalid Month (Negative Month)**:

   Input: (1, -1, 2011)

   Expected Outcome: An Error message
5. **Invalid Year (Future Year)**:

   Input: (1, 1, 2026)

   Expected Outcome: An Error message

Boundary Value Analysis Test Cases

Boundary Values for Valid Inputs:-

1. **Day Before First of the Month**:

   Input: (1, 1, 2014)

   Expected Outcome: (31, 12, 2013)

2. **Last Day of February, Non-Leap Year**:

   Input: (28, 2, 2015)

   Expected Outcome: (27, 2, 2015)

3. **Last Day of February, Leap Year**:

   Input: (29, 2, 2020)

   Expected Outcome: (28, 2, 2020)

4. **Day Boundary (31st Day)**:

   Input: (31, 10, 2017)

   Expected Outcome: (30, 10, 2017)

5. **Year Lower Boundary**:

   Input: (1, 1, 1900)

   Expected Outcome: (31, 12, 1899)

6. **Year Upper Boundary**:

   Input: (1, 1, 2015)

   Expected Outcome: (31, 12, 2014)

7. **Day Maximum for Months with 30 Days**:

Input: (30, 6, 2014)

Expected Outcome: (29, 6, 2014)

8. **Last Valid Input for Valid Year**:

   Input: (31, 7, 2011)

   Expected Outcome: (30, 7, 2011)

Equivalence Partitioning Test Cases:-

Tester Action and Input Data	Expected Outcome
(15, 7, 2012)	(14, 7, 2012)
(29, 2, 2000)	(28, 2, 2000)
(28, 2, 2015)	(27, 2, 2015)
(30, 4, 2011)	(29, 4, 2011)
(30, 9, 2017)	(29, 9, 2017)
(0, 12, 2014)	An Error message
(-1, 11, 2013)	An Error message
(1, 0, 2010)	An Error message
(1, -1, 2011)	An Error message
(1, 1, 2026)	An Error message

Boundary Value Analysis Test Cases:-

Tester Action and Input Data	Expected Outcome
(1, 1, 2014)	(31, 12, 2013)
(28, 2, 2015)	(27, 2, 2015)
(29, 2, 2020)	(28, 2, 2020)
(31, 10, 2017)	(30, 10, 2017)
(1, 1, 1900)	(31, 12, 1899)
(1, 1, 2015)	(31, 12, 2014)
(30, 6, 2014)	(29, 6, 2014)
(31, 7, 2011)	(30, 7, 2011)

b) Modify your programs such that it runs, and then execute your test suites on the program.
While executing your input data in a program, check whether the identified expected outcome
(mentioned by you) is correct or not.

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

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

string getPreviousDate(int day, int month, int year) {
    // Validate inputs
    if (year < 1900 || year > 2015 || month < 1 || month > 12 || day < 1 || day > 31) {
        return "Invalid date";
    }
```

```
    // Days in each month
    int daysInMonth[] = {31, isLeapYear(year) ? 29 : 28, 31, 30, 31, 30, 31, 31, 30,
31, 30, 31};

    // Check for the valid day in the given month
    if (day > daysInMonth[month - 1]) {
        return "Invalid date";
    }

    // Calculate previous date
    if (day > 1) {
        return to_string(day - 1) + "/" + to_string(month) + "/" + to_string(year);
    } else {
        if (month == 1) {
            // January goes to December of the previous year
            return to_string(31) + "/12/" + to_string(year - 1);
        } else {
            // Go to the last day of the previous month
            return to_string(daysInMonth[month - 2]) + "/" + to_string(month - 1) + "/"
+ to_string(year);
        }
    }
}

int main() {
    int day, month, year;

    // Input: day, month, year
    cout << "Enter day: ";
    cin >> day;
    cout << "Enter month: ";
    cin >> month;
    cout << "Enter year: ";
    cin >> year;

    // Calculate and print the previous date
```

```
    string previousDate = getPreviousDate(day, month, year);
    cout << "Previous date: " << previousDate << endl;

    return 0;
}
```

Testing the Program

Test Case Input	Expected Output
(1, 1, 2014)	"31/12/2013"
(1, 3, 2017)	"28/2/2017"
(29, 2, 2020)	"28/2/2020"
(1, 7, 2013)	"30/6/2013"
(31, 1, 2015)	"30/1/2015"
(1, 13, 2015)	"Invalid date"
(32, 1, 2015)	"Invalid date"
(1, 1, 1899)	"Invalid date"

Checking Outcomes:-

For each input, check if the output matches the expected outcome:

1. Run the program.
2. Input the day, month, and year as specified in the test cases.
3. Compare the output to the expected resultIf they match, the test case passes; if not, it fails.

Question 2

P1)

Equivalence Classes

- **Class 1**: Empty array → Output: -1
- **Class 2**: Value exists (first occurrence at index 0) → Output: 0
- **Class 3**: Value exists (first occurrence at index n, n > 0) → Output: n
- **Class 4**: Value does not exist in the array → Output: -1
- **Class 5**: Value exists with duplicates → Output: Index of the first occurrence.

Test Cases

Input (v, a)	Expected Output	Covers Equivalence Class
(5, [])	-1	1
(3, [3, 1, 2])	0	2
(4, [1, 4, 2])	1	3
(7, [1, 2, 3, 7])	3	3
(5, [1, 2, 3, 4, 5])	4	3
(6, [1, 2, 3, 4, 5])	-1	4
(3, [3, 3, 3, 4, 5])	0	5

P2)

Equivalence Classes

1. **Class 1**: Array is empty.
2. **Class 2**: The value (v) is not present in the array (a).
3. **Class 3**: The value (v) appears exactly once in the array (a).
4. **Class 4**: The value (v) appears more than once in the array (a).
5. **Class 5**: Array contains multiple different elements, and v appears exactly n times, where n > 1.

Test Cases

Input (v, a)	Expected Output	Covers Equivalence Class
(5, [])	0	1
(3, [1, 2, 4, 6])	0	2
(2, [2, 1, 4, 6])	1	3
(4, [1, 4, 4, 4, 6])	3	4
(5, [5, 5, 5, 5, 5])	5	5

P3)

Equivalence Classes

- **Class 1**: Empty array → Output: -1
- **Class 2**: Value exists at the first index → Output: Index 0
- **Class 3**: Value exists at a middle index → Output: Index of v
- **Class 4**: Value exists at the last index → Output: Index of last element
- **Class 5**: Value does not exist (less than the smallest element) → Output: -1
- **Class 6**: Value does not exist (greater than the largest element) → Output: -1
- **Class 7**: Value does not exist (between two elements) → Output: -1
- **Class 8**: Value exists with duplicates → Output: Index of any occurrence of v

Test Cases

Input (v, a)	Expected Output	Covers Equivalence Class
(5, [])	-1	1
(3, [1, 2, 3, 4])	2	3
(1, [1, 2, 3, 4])	0	2
(4, [1, 2, 3, 4])	3	4
(0, [1, 2, 3, 4])	-1	5
(5, [1, 2, 3, 4])	-1	6
(2, [1, 3, 5, 7])	-1	7

Input (v, a)	Expected Output	Covers Equivalence Class
(3, [1, 2, 3, 3, 4])	2	8

P4)

Equivalence Classes

- **Class 1**: Invalid triangle (any two sides add up to less than or equal to the third side) → Output: INVALID
- **Class 2**: Invalid triangle (negative or zero-length sides) → Output: INVALID
- **Class 3**: Equilateral triangle (all sides equal) → Output: EQUILATERAL
- **Class 4**: Isosceles triangle (two sides equal) → Output: ISOSCELES
- **Class 5**: Scalene triangle (no sides equal) → Output: SCALENE

Test Cases

Input (a, b, c)	Expected Output	Covers Equivalence Class
(1, 2, 3)	INVALID	1
(1, 1, 2)	INVALID	1
(0, 1, 1)	INVALID	2
(-1, 2, 3)	INVALID	2
(3, 3, 3)	EQUILATERAL	3
(3, 3, 2)	ISOSCELES	4
(3, 4, 5)	SCALENE	5
(2, 2, 3)	ISOSCELES	4

P5)

Equivalence Classes

- **Class 1**: s1 is a prefix of s2 → Output: true
- **Class 2**: s1 is not a prefix of s2 → Output: false
- **Class 3**: s1 is empty → Output: true (empty string is a prefix of any string)

- **Class 4**: s1 is longer than s2 → Output: false
- **Class 5**: s1 equals s2 (both strings are identical) → Output: true

Test Cases

Input (s1, s2)	Expected Output	Covers Equivalence Class
("abc", "abcdef")	true	1
("abc", "abXdef")	false	2
("", "abcdef")	true	3
("abcdefg", "abc")	false	4
("abc", "abc")	true	5

P6)

a) Identify the Equivalence Classes

- **Class 1**: Equilateral triangle (A = B = C).
- **Class 2**: Isosceles triangle (two sides equal, A = B or A = C or B = C).
- **Class 3**: Scalene triangle (no sides equal, A ≠ B ≠ C).
- **Class 4**: Right-angled triangle ($A^2 + B^2 = C^2$ or any permutation).
- **Class 5**: Not a triangle (the sum of two sides is less than or equal to the third side).
- **Class 6**: Non-positive values for sides (A ≤ 0 or B ≤ 0 or C ≤ 0).

b) Identify Test Cases to Cover the Equivalence Classes

Input (A, B, C)	Expected Output	Covers Equivalence Class
(3, 3, 3)	Equilateral	1
(4, 4, 5)	Isosceles	2
(3, 4, 5)	Right-angled	4
(5, 7, 8)	Scalene	3
(1, 2, 3)	Not a triangle	5
(0, 3, 4)	Not a triangle	6

Input (A, B, C)	Expected Output	Covers Equivalence Class
(-1, 2, 3)	Not a triangle	6

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

Input (A, B, C)	Expected Output	Boundary Condition Covered
(3, 4, 7)	Not a triangle	Boundary of sum close to C

d) Boundary Test Cases for A = C (Isosceles Triangle)

Input (A, B, C)	Expected Output	Boundary Condition Covered
(5, 7, 5)	Isosceles	Boundary A = C
(5, 5, 7)	Isosceles	Boundary A = B

e) Boundary Test Cases for A = B = C (Equilateral Triangle)

Input (A, B, C)	Expected Output	Boundary Condition Covered
(5, 5, 5)	Equilateral	Boundary A = B = C

f) Boundary Test Cases for $A^2 + B^2 = C^2$ (Right-Angle Triangle)

Input (A, B, C)	Expected Output	Boundary Condition Covered
(3, 4, 5)	Right-angled	Boundary $A^2 + B^2 = C^2$
(6, 8, 10)	Right-angled	Boundary $A^2 + B^2 = C^2$

g) Boundary Test Cases for Non-Triangle Case

Input (A, B, C)	Expected Output	Boundary Condition Covered
(1, 2, 3)	Not a triangle	Boundary A + B = C
(1, 2., 5)	Not a triangle	Boundary A + B < C

h) Boundary Test Cases for Non-Positive Inputs

Input (A, B, C)	Expected Output	Boundary Condition Covered
(0, 2, 3)	Not a triangle	Boundary A = 0
(-1, 2, 3)	Not a triangle	Boundary A < 0
(2, 0, 3)	Not a triangle	Boundary B = 0
(3, 4, 0)	Not a triangle	Boundary C = 0