Name: Vasoya Neel Naresbhai
ID:202201448

Q1-Test Suite for Previous Date Program
ANS:
1. Equivalence Partitioning

Valid Equivalence Classes:
1. Day: 1-31
2. Month: 1-12
3. Year: 1900-2015

Invalid Equivalence Classes:
4. Day: < 1 or > 31
5. Month: < 1 or > 12
6. Year: < 1900 or > 2015

Test Cases:
1. (15, 6, 2000) - Expected: (14, 6, 2000)
2. (1, 7, 1950) - Expected: (30, 6, 1950)
3. (1, 1, 2010) - Expected: (31, 12, 2009)
4. (1, 3, 2000) - Expected: (29, 2, 2000) [Leap year]
5. (1, 3, 2001) - Expected: (28, 2, 2001) [Non-leap year]
6. (32, 5, 2000) - Expected: Invalid date
7. (15, 13, 2000) - Expected: Invalid date
8. (15, 6, 1899) - Expected: Invalid date

2. Boundary Value Analysis
Test Cases:
1. (1, 1, 1900) - Expected: Invalid date [Lower bound]
2. (31, 12, 2015) - Expected: (30, 12, 2015) [Upper bound]
3. (1, 1, 1901) - Expected: (31, 12, 1900)
4. (31, 12, 2014) - Expected: (30, 12, 2014)
5. (1, 1, 2000) - Expected: (31, 12, 1999)
6. (31, 1, 2000) - Expected: (30, 1, 2000)
7. (1, 2, 2000) - Expected: (31, 1, 2000)
8. (29, 2, 2000) - Expected: (28, 2, 2000) [Leap year]
9. (1, 3, 2000) - Expected: (29, 2, 2000) [Leap year]
10. (1, 3, 2001) - Expected: (28, 2, 2001) [Non-leap year]
11. (31, 12, 1900) - Expected: (30, 12, 1900)
12. (1, 1, 2016) - Expected: Invalid date
13. (0, 6, 2000) - Expected: Invalid date
14. (32, 6, 2000) - Expected: Invalid date
15. (15, 0, 2000) - Expected: Invalid date
16. (15, 13, 2000) - Expected: Invalid date

C++ code

```
bool is_leap_year(int year) {
    return (year % 4 == 0 && (year % 100 != 0 || year % 400 == 0));
}
string previous_date(int day, int month, int year) {
    if (!(1 <= month && month <= 12 && 1 <= day && day <= 31 && 1900 <= year && year <=
2015)) {
        return "Invalid date";
    }

    int days_in_month[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

    if (is_leap_year(year)) {
        days_in_month[1] = 29;
    }

    if (day > days_in_month[month - 1]) {
        return "Invalid date";
    }

    if (day > 1) {
        return to_string(day - 1) + "/" + to_string(month) + "/" + to_string(year);
    } else if (month > 1) {
        return to_string(days_in_month[month - 2]) + "/" + to_string(month - 1) + "/" +
to_string(year);
    } else {
        return (year > 1900) ? "31/12/" + to_string(year - 1) : "Invalid date";
    }
}
int main() {
    int day = 1, month = 1, year = 2000;
    cout << previous_date(day, month, year) << endl;
    return 0;
}
```

USE OF CODE:

```
print(previous_date(15, 6, 2000))
        Output: (14, 6, 2000)
print(previous_date(1, 1, 1900))
        Output: Invalid date
```

QUESTION 2:

P1: linearSearch

Equivalence Partitioning:
1. v exists in a
2. v does not exist in a
3. a is empty

Boundary Value Analysis:
4. v is the first element of a
5. v is the last element of a
6. a has only one element (v)
7. a has only one element (not v)

Test Cases:
1. (5, [1, 3, 5, 7, 9]) → 2
2. (6, [1, 3, 5, 7, 9]) → -1
3. (5, []) → -1
4. (1, [1, 3, 5, 7, 9]) → 0
5. (9, [1, 3, 5, 7, 9]) → 4
6. (5, [5]) → 0
7. (6, [5]) → -1

P2: countItem

Equivalence Partitioning:
1. v exists in a multiple times
2. v exists in a once
3. v does not exist in a
4. a is empty

Boundary Value Analysis:
5. All elements in a are v
6. Only the first element is v
7. Only the last element is v

Test Cases:
1. (5, [1, 5, 2, 5, 5, 3]) → 3
2. (5, [1, 2, 3, 4, 5]) → 1
3. (6, [1, 2, 3, 4, 5]) → 0
4. (5, []) → 0
5. (5, [5, 5, 5, 5, 5]) → 5
6. (5, [5, 1, 2, 3, 4]) → 1
7. (5, [1, 2, 3, 4, 5]) → 1

P3: binarySearch

Equivalence Partitioning:
1. v exists in a
2. v does not exist in a
3. a is empty

Boundary Value Analysis:
4. v is the first element of a
5. v is the last element of a
6. v is in the middle of a
7. a has only one element (v)
8. a has only one element (not v)
9. v is less than all elements in a
10. v is greater than all elements in a

Test Cases:
1. (5, [1, 3, 5, 7, 9]) → 2
2. (6, [1, 3, 5, 7, 9]) → -1
3. (5, []) → -1
4. (1, [1, 3, 5, 7, 9]) → 0
5. (9, [1, 3, 5, 7, 9]) → 4
6. (5, [1, 3, 5, 7, 9]) → 2
7. (5, [5]) → 0
8. (6, [5]) → -1
9. (0, [1, 3, 5, 7, 9]) → -1
10. (10, [1, 3, 5, 7, 9]) → -1

P4: triangle

Equivalence Partitioning:
1. Equilateral triangle
2. Isosceles triangle
3. Scalene triangle
4. Invalid triangle

Boundary Value Analysis:
5. a + b = c (invalid)
6. a + b = c + 1 (valid scalene)
7. a = b, b ≠ c (isosceles)
8. a = b = c (equilateral)

Test Cases:
1. (5, 5, 5) → EQUILATERAL
2. (5, 5, 6) → ISOSCELES
3. (3, 4, 5) → SCALENE
4. (1, 2, 3) → INVALID
5. (5, 5, 10) → INVALID

6. (5, 5, 9) → SCALENE
7. (5, 5, 7) → ISOSCELES
8. (6, 6, 6) → EQUILATERAL

P5: prefix

Equivalence Partitioning:
1. s1 is a prefix of s2
2. s1 is not a prefix of s2
3. s1 is empty
4. s2 is empty
5. s1 is longer than s2

Boundary Value Analysis:
6. s1 equals s2
7. s1 differs from s2 in the last character
8. s1 differs from s2 in the first character

Test Cases:
1. ("hello", "helloworld") → true
2. ("hi", "hello") → false
3. ("", "hello") → true
4. ("hello", "") → false
5. ("hello", "hi") → false
6. ("hello", "hello") → true
7. ("hell", "hello") → true
8. ("hella", "hello") → false

P6: Triangle Classification (Floating-Point Version)

a) Equivalence Classes

1. Valid triangles:
   - EC1: Scalene triangle (all sides different)
   - EC2: Isosceles triangle (two sides equal)
   - EC3: Equilateral triangle (all sides equal)
   - EC4: Right-angled triangle

2. Invalid triangles:
   - EC5: Non-triangle (sum of two sides ≤ third side)
   - EC6: Non-positive input (any side ≤ 0)

b) Test Cases for Equivalence Classes

1. TC1 (EC1): (3.0, 4.0, 6.0) - Scalene triangle
2. TC2 (EC2): (5.0, 5.0, 7.0) - Isosceles triangle
3. TC3 (EC3): (6.0, 6.0, 6.0) - Equilateral triangle

4. TC4 (EC4): (3.0, 4.0, 5.0) - Right-angled triangle
5. TC5 (EC5): (1.0, 2.0, 3.0) - Non-triangle
6. TC6 (EC6): (0.0, 4.0, 5.0) - Non-positive input

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

7. TC7: (3.0, 4.0, 6.99) - Just valid
8. TC8: (3.0, 4.0, 7.0) - Boundary condition
9. TC9: (3.0, 4.0, 7.01) - Just invalid

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

10. TC10: (5.0, 4.0, 5.0) - Isosceles with A = C
11. TC11: (5.0, 4.99, 5.0) - Just isosceles
12. TC12: (5.0, 5.01, 5.0) - Just scalene

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

13. TC13: (5.0, 5.0, 5.0) - Equilateral
14. TC14: (5.0, 5.0, 4.99) - Just not equilateral
15. TC15: (5.0, 5.0, 5.01) - Just not equilateral

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

16. TC16: (3.0, 4.0, 5.0) - Right-angled
17. TC17: (3.0, 4.0, 4.99) - Just not right-angled
18. TC18: (3.0, 4.0, 5.01) - Just not right-angled

g) Boundary Cases for Non-triangle

19. TC19: (1.0, 2.0, 3.0) - Sum of two sides equal to third side
20. TC20: (1.0, 1.0, 2.0) - Sum of two sides equal to third side
21. TC21: (0.9, 2.0, 3.0) - Sum of two sides less than third side

h) Test Points for Non-positive Input

22. TC22: (0.0, 4.0, 5.0) - Zero input
23. TC23: (-1.0, 4.0, 5.0) - Negative input
24. TC24: (0.0, 0.0, 0.0) - All zero input