IT 314 LAB 8
202201285 - Madhav Jivani

Q1)

In this problem, we need to determine the previous date based on a given input of day, month, and year. The solution involves designing test cases using Equivalence Partitioning (EP) and Boundary Value Analysis (BVA) to identify both valid and invalid input ranges.

1. Equivalence Partitioning (EP)

In Equivalence Partitioning, we divide the input data into equivalent classes, where each class represents a set of valid or invalid input values. This way, we can minimize the number of test cases while still covering the entire input range.

Equivalence Classes for Day, Month, and Year:

- Day:
  - Valid: $1 \leq day \leq 31$
  - Invalid: $day < 1$, $day > 31$
- Month:
  - Valid: $1 \leq month \leq 12$
  - Invalid: $month < 1$, $month > 12$
- Year:
  - Valid: $1900 \leq year \leq 2015$
  - Invalid: $year < 1900$, $year > 2015$

Equivalence Partitioning Test Cases:

Test Case ID	Day	Month	Year	Expected Outcome
EP-01	15	6	2010	Previous date: 14/06/2010
EP-02	1	3	2012	Previous date: 29/02/2012 (leap year)
EP-03	1	1	2000	Previous date: 31/12/1999
EP-04	32	5	2010	Invalid date (day out of range)
EP-05	15	13	2011	Invalid date (month out of range)
EP-06	25	5	1885	Invalid date (year out of range)

Boundary Value Analysis Test Cases:

Test Case ID	Day	Month	Year	Expected Outcome
BVA-01	1	1	1900	Previous date: 31/12/1899
BVA-02	0	5	2012	Invalid date (day = 0)
BVA-03	31	12	2015	Previous date: 30/12/2015
BVA-04	32	10	2000	Invalid date (day = 32)
BVA-05	15	0	1999	Invalid date (month = 0)
BVA-06	15	12	2016	Invalid date (year = 2016)

CODE : (image)

```python
from datetime import datetime, timedelta

def get_previous_date(day, month, year):
    try:
        # Create a date object
        current_date = datetime(year, month, day)
        # Subtract one day to get the previous date
        previous_date = current_date - timedelta(days=1)
        # Return the previous date in DD/MM/YYYY format
        return previous_date.strftime('%d/%m/%Y')
    except ValueError:
        return "Invalid date"

# Test cases from EP and BVA
test_cases = [
    (15, 6, 2010),  # EP-01
    (1, 3, 2012),   # EP-02
    (1, 1, 2000),   # EP-03
    (32, 5, 2010),  # EP-04
    (15, 13, 2011), # EP-05
    (25, 5, 1885),  # EP-06
    (1, 1, 1900),   # BVA-01
    (0, 5, 2012),   # BVA-02
    (31, 12, 2015), # BVA-03
    (32, 10, 2000), # BVA-04
    (15, 0, 1999),  # BVA-05
    (15, 12, 2016), # BVA-06
]

# Execute the test cases
for day, month, year in test_cases:
    result = get_previous_date(day, month, year)
    print(f"Input: Day={day}, Month={month}, Year={year} => Output: {result}")
```

Test Cases for Previous Date Program

Test Case ID	Day	Month	Year	Expected Outcome	Reason
TC-01	15	6	2010	Previous date: 14/06/2010	Valid input, standard date.
TC-02	29	2	2012	Previous date: 28/02/2012	Valid input, leap year (29 Feb).
TC-03	1	1	2000	Previous date: 31/12/1999	Valid input, year change.
TC-04	32	5	2010	Invalid date	Day exceeds the maximum limit (31).
TC-05	15	13	2011	Invalid date	Month exceeds the maximum limit (12).
TC-06	25	5	1885	Invalid date	Year is below the minimum limit (1900).
TC-07	0	5	2012	Invalid date	Day is less than the minimum limit (1).
TC-08	15	0	1999	Invalid date	Month is less than the minimum limit (1).

TC-09	15	12	2016	Invalid date	Year exceeds the maximum limit (2015).
TC-10	31	4	2015	Previous date: 30/04/2015	Valid input, testing end of month.
TC-11	1	3	1900	Previous date: 28/02/1900	Valid input, 1900 is not a leap year.
TC-12	31	2	2015	Invalid date	February has only 28 days in 2015 (not leap).

Q2) ->

P1)

Test Cases

Test Case ID	Input Value (v)	Input Array (a)	Expected Output	Reason
TC-01	5	[1, 2, 3, 4, 5]	4	Value is found at index 4 (0-based indexing).
TC-02	1	[1, 2, 3, 4, 5]	0	Value is found at index 0 (first element).
TC-03	10	[1, 2, 3, 4, 5]	-1	Value not found in the array.
TC-04	3	[3, 3, 3, 3]	0	Value found at index 0 (first occurrence).
TC-05	2	[4, 2, 2, 4]	1	Value found at index 1 (first occurrence).
TC-06	7	[]	-1	Value not found in an empty array.
TC-07	0	[-1, 0, 1]	1	Value is found at index 1.

TC-08	99	[1, 2, 3, 4, 5, 99]	5	Value is found at index 5 (last element).
TC-09	4	[4, 4, 4, 4, 4]	0	Value found at index 0 (all elements same).
TC-10	-5	[1, -2, -5, 4]	2	Value found at index 2.

P2)

Test Cases

Test Case ID	Input Value (v)	Input Array (a)	Expected Output	Reason
TC-01	5	[1, 2, 3, 4, 5]	1	Value appears once in the array.
TC-02	3	[1, 2, 3, 4, 5]	1	Value appears once, present in the middle.
TC-03	10	[1, 2, 3, 4, 5]	0	Value not found in the array.
TC-04	3	[3, 3, 3, 3]	4	Value appears multiple times (all elements).
TC-05	2	[4, 2, 2, 4]	2	Value appears twice.
TC-06	0	[]	0	Value not found in an empty array.
TC-07	1	[-1, 0, 1]	1	Value is found once in the array.
TC-08	99	[1, 2, 3, 4, 5, 99]	1	Value appears once at the end.

TC-09	4	[4, 4, 4, 4, 4]	5	Value found at all positions in the array.
TC-10	-5	[1, -2, -5, 4]	1	Value appears once (negative number).

P3)

Test Cases

Test Case ID	Input Value (v)	Input Array (a)	Expected Output	Reason
TC-01	5	[1, 2, 3, 4, 5]	4	Value is present at the end of the array.
TC-02	1	[1, 2, 3, 4, 5]	0	Value is the first element.
TC-03	10	[1, 2, 3, 4, 5]	-1	Value not found in the array.
TC-04	3	[1, 2, 3, 4, 5]	2	Value found in the middle of the array.
TC-05	4	[1, 2, 3, 4, 5]	3	Value found just before the last element.
TC-06	0	[1, 2, 3, 4, 5]	-1	Value less than the minimum in the array.
TC-07	2	[1, 1, 2, 2, 3, 4]	2	Value found at index 2 (first occurrence).
TC-08	6	[1, 2, 3, 4, 5, 6]	5	Value is the last element in the array.

TC-09	-1	[-3, -2, -1, 0, 1, 2]	2	Value found in the positive numbers.
TC-10	5	[5, 5, 5, 5, 5]	0	Value appears multiple times, should return 0.
TC-11	7	[]	-1	Search in an empty array returns -1.

P4)

Test Cases

Test Case ID	Side a	Side b	Side c	Expected Output	Reason
TC-01	3	3	3	0	All sides equal (equilateral).
TC-02	5	5	3	1	Two sides equal (isosceles).
TC-03	4	5	6	2	All sides different (scalene).
TC-04	2	2	4	3	Fails triangle inequality (invalid).
TC-05	1	2	3	3	Fails triangle inequality (invalid).
TC-06	0	5	5	3	Non-positive side length (invalid).
TC-07	-1	1	1	3	Negative side length (invalid).
TC-08	10	1	1	3	Fails triangle inequality (invalid).

TC-09	5	4	5	1	Two sides equal (isosceles).
TC-10	6	8	10	2	All sides different (scalene).
TC-11	7	3	4	3	Fails triangle inequality (invalid).

P5)

Test Cases

Test Case ID	Input s1	Input s2	Expected Output	Reason
TC-01	"abc"	"abcdef"	true	s1 is a prefix of s2.
TC-02	"abc"	"abcde"	true	s1 matches the beginning of s2.
TC-03	"abcd"	"abc"	false	s1 is longer than s2.
TC-04	"abc"	"ab"	false	s1 is longer than s2.
TC-05	""	"abc"	true	An empty string is always a prefix.
TC-06	"abc"	""	false	Non-empty s1 cannot be a prefix of an empty s2.
TC-07	"abc"	"aBcdef"	false	Case-sensitive check (mismatch).
TC-08	"a"	"abc"	true	Single character prefix match.
TC-09	"abcdef"	"abcdef"	true	Both strings are equal, so s1 is a prefix.

TC-10	"abcdef"	"abcde"	false	s1 is longer than s2.
TC-11	"abc"	"xyzabc"	false	s1 does not match the beginning of s2.

P6)

a) Identify the Equivalence Classes

The equivalence classes for the triangle classification system can be divided into valid and invalid categories based on the input sides (A, B, C):

Valid Classes

1. Equilateral Triangle: A=B=CA = B = CA=B=C (e.g., 3,3,33, 3, 33,3,3)
2. Isosceles Triangle: A=B≠CA = B \neq CA=B=C or A=C≠BA = C \neq BA=C=B or B=C≠AB = C \neq AB=C=A (e.g., 3,3,43, 3, 43,3,4)
3. Scalene Triangle: A≠B≠CA \neq B \neq CA=B=C and satisfies A+B>CA + B > CA+B>C, A+C>BA + C > BA+C>B, and B+C>AB + C > AB+C>A (e.g., 3,4,53, 4, 53,4,5)
4. Right-Angled Triangle: A2+B2=C2A^2 + B^2 = C^2A2+B2=C2 (with appropriate ordering of sides, e.g., 3,4,53, 4, 53,4,5 or 5,3,45, 3, 45,3,4)

Invalid Classes

5. Non-Triangle: A+B≤CA + B \leq CA+B≤C or A+C≤BA + C \leq BA+C≤B or B+C≤AB + C \leq AB+C≤A (e.g., 1,1,31, 1, 31,1,3)
6. Non-positive Sides: At least one of the values AAA, BBB, or CCC is non-positive (e.g., −1,2,3-1, 2, 3−1,2,3 or 0,2,30, 2, 30,2,3)

b) Identify Test Cases to Cover the Identified Equivalence Classes

Test Case ID	Input Values (A, B, C)	Expected Output	Equivalence Class
TC-01	3, 3, 3	Equilateral	Equilateral Triangle
TC-02	3, 3, 4	Isosceles	Isosceles Triangle
TC-03	3, 4, 5	Scalene	Scalene Triangle
TC-04	5, 3, 4	Right Angled	Right-Angled Triangle
TC-05	1, 1, 3	Not a triangle	Non-Triangle
TC-06	2, 2, -1	Non-positive	Non-positive Input
TC-07	0, 2, 3	Non-positive	Non-positive Input
TC-08	-1, 5, 5	Non-positive	Non-positive Input

Explanation of Test Cases

- TC-01 covers the Equilateral Triangle equivalence class.
- TC-02 covers the Isosceles Triangle equivalence class.
- TC-03 covers the Scalene Triangle equivalence class.
- TC-04 covers the Right-Angled Triangle equivalence class.

- TC-05 covers the Non-Triangle equivalence class.
- TC-06, TC-07, and TC-08 cover the Non-positive Input equivalence class.

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

To verify the boundary for the scalene triangle (A + B > C), we can use the following test cases:

Test Case ID	Input Values (A, B, C)	Expected Output	Reason
TC-09	3, 4, 5	Scalene	A + B > C (7 > 5)
TC-10	3, 4, 6	Scalene	A + B > C (7 > 6)
TC-11	2, 2, 3	Not a triangle	A + B = C (4 = 3)
TC-12	2, 2, 4	Not a triangle	A + B < C (4 < 4)

Explanation of Test Cases

- TC-09 and TC-10 test valid scalene triangle cases where the sum of two sides is greater than the third.
- TC-11 tests the boundary where the sum equals the third side (non-triangle).
- TC-12 tests the condition where the sum is less than the third side (non-triangle).

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

To verify the boundary condition for the isosceles triangle (A = C), we can use the following test cases:

Test Case ID	Input Values (A, B, C)	Expected Output	Reason
TC-13	3, 4, 3	Isosceles	A = C (3 = 3)
TC-14	5, 2, 5	Isosceles	A = C (5 = 5)
TC-15	2, 2, 3	Isosceles	A = B and A < C (2 = 2)
TC-16	0, 2, 0	Non-positive	Non-positive sides (0 is not valid)

Explanation of Test Cases

- TC-13 and TC-14 confirm that when A and C are equal, the triangle is classified as isosceles.
- TC-15 also confirms isosceles classification while ensuring A < C.
- TC-16 checks for non-positive input conditions.

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

To verify the boundary condition for the equilateral triangle (A = B = C), we can use the following test cases:

Test Case ID	Input Values (A, B, C)	Expected Output	Reason
TC-17	5, 5, 5	Equilateral	All sides are equal (5 = 5 = 5)
TC-18	0, 0, 0	Non-positive	All sides are zero (invalid input)
TC-19	-1, -1, -1	Non-positive	All sides negative (invalid input)
TC-20	1, 1, 1	Equilateral	Smallest valid equilateral triangle

Explanation of Test Cases

- TC-17 and TC-20 verify that when all sides are equal, the triangle is classified as equilateral.
- TC-18 and TC-19 check for non-positive conditions.

f) Boundary Condition for $A^2 + B^2 = C^2$ (Right-Angled Triangle)

To verify the boundary condition for the right-angled triangle ($A^2 + B^2 = C^2$), we can use the following test cases:

Test Case ID	Input Values (A, B, C)	Expected Output	Reason
TC-21	3, 4, 5	Right Angled	$3^2 + 4^2 = 5^2$ (9 + 16 = 25)
TC-22	5, 12, 13	Right Angled	$5^2 + 12^2 = 13^2$ (25 + 144 = 169)
TC-23	1, 1, √2	Right Angled	$1^2 + 1^2 = (\sqrt{2})^2$ (1 + 1 = 2)
TC-24	6, 8, 10	Right Angled	$6^2 + 8^2 = 10^2$ (36 + 64 = 100)

Explanation of Test Cases

- TC-21 and TC-22 confirm valid right-angled triangle cases.
- TC-23 uses floating-point representation to check for a right angle.
- TC-24 is another right-angled triangle example.

g) Non-Triangle Case (Boundary)

To explore the boundaries for non-triangle cases, we can use the following test cases:

Test Case ID	Input Values (A, B, C)	Expected Output	Reason
TC-25	1, 1, 2	Not a triangle	A + B = C (1 + 1 = 2)
TC-26	1, 2, 3	Not a triangle	A + B = C (1 + 2 = 3)
TC-27	0, 1, 2	Not a triangle	One side is zero (invalid)
TC-28	2, 1, 0	Not a triangle	One side is zero (invalid)

Explanation of Test Cases

- TC-25 and TC-26 test the boundaries where the sum of two sides is equal to the third side (non-triangle).
- TC-27 and TC-28 include cases where sides are non-positive.

h) Non-Positive Input Test Points

For non-positive input conditions, the following test cases can be utilized:

Test Case ID	Input Values (A, B, C)	Expected Output	Reason
TC-29	-1, 5, 5	Non-positive	One side is negative
TC-30	0, 3, 4	Non-positive	One side is zero
TC-31	5, 0, 2	Non-positive	One side is zero
TC-32	-3, -4, -5	Non-positive	All sides are negative

Explanation of Test Cases

- TC-29, TC-30, TC-31, and TC-32 focus on cases where at least one of the triangle sides is non-positive.