

**IT314**

**Lab-08**

Name: Stavan Ravisaheb

ID: 202201436

Q.1. Equivalence Class Test Cases for Previous Date Program

Problem:

A program takes three inputs: day, month, and year. We need to determine the previous date or check for invalid input based on specific ranges:

1 ≤ day ≤ 31

1 ≤ month ≤ 12

1900 ≤ year ≤ 2015

We need to design equivalence class test cases using Equivalence Partitioning and Boundary Value Analysis.

Equivalence Class Partitioning

|  |  |
| --- | --- |
| Tester Action & Input Data | Expected Outcome |
| 32, 12, 2000 | Error message (Invalid date) date) |
| 29, 2, 2021 | Error message (Invalid year) |
| 15, 13, 2005 | Error message (Invalid month) |
| 31, 4, 2001 | Error message (Invalid day for month) |
| 1, 1, 1900 | 31, 12, 1899 (Boundary test) |
| 15, 8, 2000 | 14, 8, 2000 (Valid date) |
| 1, 3, 2001 | 28, 2, 2001 (Non-leap year transition) |
| 1, 3, 2000 | 29, 2, 2000 (Leap year transition) |

Boundary Value Analysis

|  |  |
| --- | --- |
| Tester Action and Input Data | Expected Outcome |
| 1, 1, 1900 | Error message (Year too low) |
| 31, 12, 2015 | Error message (Year too high) |
| 0, 5, 2010 | Error message (Invalid day) |
| 32, 1, 2010 | Error message (Invalid day) |
| 1, 0, 2010 | Error message (Invalid month) |
| 1, 13, 2010 | Error message (Invalid month) |

Q.2. Program Test Cases

P1: Linear Search

The linearSearch function takes a value v and an array a[] and returns the index of v if found, or -1 otherwise.

Test Cases for Linear Search:

|  |  |
| --- | --- |
| Input Data | Expected Outcome |
| v=5, a[]={1, 2, 5, 6} | 2 (Value found at index 2) |
| v=9, a[]={1, 2, 5, 6} | -1 (Value not found) |
| v=3, a[]={3, 3, 3} | 0 (First occurence at index 0) |
| v=3, a[]={} | -1 (empty array) |

P2: Count Item

The countItem function counts the number of times v appears in the array a[].

Test Cases for Count Item:

|  |  |
| --- | --- |
| Input Data | Expected Outcome |
| v=2, a[]={2, 2, 3, 4} | 2 (Value appears twice) |
| v=5, a[]={1, 2, 3, 4} | 0 (Value not present) |
| v=3, a[]={3, 3, 3} | 3 (Value appears three times) |
| v=4, a[]={} | 0 (Empty array) |

P3: Binary Search

The binarySearch function searches for a value v in a sorted array a[] using binary search.

Test Cases for Binary Search:

|  |  |
| --- | --- |
| Input Data | Expected Outcome |
| v=6, a[]={1, 3, 5, 6, 9} | 3 (Value found at index 3) |
| v=7, a[]={1, 3, 5, 6 9} | -1 (Value not found) |
| v=2, a[]={2} | 0 (Only one element) |
| v=4, a[]={} | -1 (Empty array) |

P4: Triangle Classification

The triangle function determines if three side lengths form an equilateral, isosceles, scalene, or invalid triangle.

Test Cases for Triangle Classification:

|  |  |
| --- | --- |
| Input Data | Expected Outcome |
| 3, 4, 5 | Scalene |
| 2, 2, 2 | Equilateral |
| 3 ,3 4 | Isosceles |
| 1, 2, 3 | Invalid |

P5: Prefix Check

The prefix function checks whether string s1 is a prefix of string s2.

Test Cases for Prefix Check:

|  |  |
| --- | --- |
| Input Data | Expected Outcome |
| s1=”abc”, s2=”abcdef” | True |
| s1=”abc”, s2=”ab” | False |
| s1=”a”, s2=”a” | True |
| s1=”xyz”, s2=”xyabcdef” | False |

P6: Triangle Classification with Floating Point Values

This problem is similar to P4 but uses floating-point numbers for the sides and includes a right-angle check.

Test Cases for P6:

|  |  |
| --- | --- |
| Input Data | Expected Outcome |
| 3.0, 4.0, 5.0 | Right-angled |
| 3.0, 3.0, 3.0 | Equilateral |
| 2.0, 2.0, 3.0 | Isosceles |
| 1.0, 2.0, 3.0 | Invalid |

---