Lab1 - Verification & Validation - Test Cases

Nguyễn Việt Đức – 3122411044 – DCT122C5

# I. Verification and Validation

Problem: Solve the quadratic equation ax² + bx + c = 0.  
**What are the problems of those two systems? Write down your answer here.**

* **System 1:** The formula for x2 = -b - sqrt(DELTA/2a) is incorrect. The correct quadratic formula should be x = [-b ± sqrt(DELTA)] / (2a), so x2 should be (-b - sqrt(DELTA)) / (2a). This error will produce incorrect results for x2. Additionally, DELTA is used without being explicitly calculated or validated, which could lead to errors if DELTA is negative (complex roots) or zero (repeated root), though extreme cases are excluded per spec.
* **System 2:** The system correctly calculates DELTA first and uses the proper quadratic formula for both x1 and x2. The step-by-step approach allows for DELTA verification, enhancing validation. However, it still lacks handling for DELTA < 0 (no real roots) or DELTA = 0 (one root), which could cause issues if such inputs are encountered, despite the spec ignoring extreme cases.

# II. Test Cases

1. **Function f1(int x):**

We need 2 test cases to cover both branches (x > 10 and x ≤ 10).

* Test case 1: x = 11, expected output = 22
* Test case 2: x = 5, expected output = -5

**b) Function f1 implemented incorrectly:**

Using the test cases from (a):

* x = 11: Returns 22 (matches original, no error detected)
* x = 5: Returns -5 (matches original, no error detected)

These test cases cannot detect the error because they don’t cover x ≤ 0, where the behavior differs. For example, with x = -5: the original returns 5, but the implemented version returns -10 (error).

**In this case, how many test-cases we need to test this function? What are they?**

* We need 3 test cases to cover all branches (x > 10, 0 < x ≤ 10, x ≤ 0).
  + Test case 1: x = 11 (> 10), expected = 22
  + Test case 2: x = 5 (0 < x ≤ 10), expected = -5
  + Test case 3: x = -5 (x ≤ 0), expected = -10

**c) Function f2(int x):**

The condition "else if (x < 2)" is unreachable because if x ≥ 10, x cannot be < 2. The function effectively returns 2 \* x for all x. We need 2 test cases for the reachable branches.

* Test case 1: x = 5 (< 10), expected = 10
* Test case 2: x = 15 (≥ 10), expected = 30

**d) Function f3(int x):**

Both branches return 2 \* x, but to cover the condition, we need 2 test cases where the condition is true or false, ensuring the domain (x² \* cos(x) > 0 for log).

* Test case 1: x = 1 (log(1 \* cos(1)) ≈ -0.615 < 3, true), expected = 2
* Test case 2: x = -1 (log(1 \* cos(-1)) ≈ -0.615 > -3? Adjust: x = 0.1, log(0.01 \* cos(0.1)) ≈ -4.61 < 0.3, true; need false case, but both return 2\*x, so coverage is minimal).
* Practically, 1 test case suffices since outputs are identical, but for condition coverage:
  + Test case 1: x = 1, expected = 2
  + Test case 2: x = -1, expected = -2 (check condition variability)

**e) Function findMax(num1,num2,num3):**

To fully test the function findMax(int num1, int num2, int num3), we need 6 test-cases. These include:

1. Unique maximum – num1 is the largest
   * Input: (5, 3, 4)
   * Expected output: 5
   * Result: Correct.
2. Unique maximum – num2 is the largest
   * Input: (3, 5, 4)
   * Expected output: 5
   * Result: Correct.
3. Unique maximum – num3 is the largest
   * Input: (3, 4, 5)
   * Expected output: 5
   * Result: Correct.
4. Tie case – num1 and num2 are equal and largest
   * Input: (5, 5, 4)
   * Expected output: 5
   * Actual code output: 0 (error, because conditions use strict > only).
5. All numbers equal
   * Input: (5, 5, 5)
   * Expected output: 5
   * Actual code output: 0 (error, same reason as above).
6. All negative numbers
   * Input: (-1, -2, -3)
   * Expected output: -1
   * Actual code output: 0 (error, because max is initialized to 0).

# III. Practice 1

**Problem: Solve the biquadratic equation ax⁴ + bx² + c = 0.**

Input: a, b, c (real numbers).  
Output: Real solutions of the equation (0, 2, or 4 roots).

Test-cases:

1. a=0, b=0, c=0 → infinite solutions.

2. a=0, b=0, c≠0 → no solution.

3. a=0, b≠0, c≠0 with b and c having the same sign → no solution.

4. a=0, b≠0, c≠0 with b and c having opposite signs → 2 solutions.

5. a≠0, Δ < 0 → no solution.

6. a≠0, Δ = 0, solution y<0 → no solution.

7. a≠0, Δ = 0, solution y≥0 → 2 solutions.

8. a≠0, Δ > 0, both y1,y2 ≥ 0 → 4 solutions.

9. a≠0, Δ > 0, only one y ≥ 0 → 2 solutions.