**KIỂM THỬ PHẦN MỀM – LAB1**

# I. VERIFICATION AND VALIDATION

## Vấn đề của SYSTEM 1:

1. **Lỗi công thức toán học:** 
   * Công thức x2 = -b - sqrt(DELTA/2a) là sai
   * Đúng phải là x2 = (-b - sqrt(DELTA))/(2a)
   * Hiện tại hệ thống tính sqrt(DELTA/2a) thay vì sqrt(DELTA)/(2a)
2. **Thiếu kiểm tra điều kiện:** 
   * Không kiểm tra a ≠ 0 (nếu a = 0 thì không phải phương trình bậc 2)
   * Không kiểm tra DELTA ≥ 0 (nếu DELTA < 0 thì vô nghiệm trong số thực)
3. **Giao diện không rõ ràng:** 
   * Chỉ có một nút "Press here to get solutions" mà không hiển thị quá trình tính toán
   * Người dùng không biết được discriminant (DELTA) là bao nhiêu

## Vấn đề của SYSTEM 2:

1. **Thiếu kiểm tra điều kiện:** 
   * Không kiểm tra a ≠ 0
   * Không kiểm tra DELTA ≥ 0 trước khi tính căn bậc hai
2. **Vấn đề giao diện:** 
   * Chia thành 2 bước nhưng có thể gây rối cho người dùng
   * Nếu DELTA < 0, hệ thống vẫn sẽ cố tính toán và báo lỗi

# II. TEST-CASES

**Câu a:**

Số lượng test-case cần thiết: 3 test-case

* Test case 1: x > 10 (ví dụ: x = 15)
  + Input: 15, Expected: 30 (2 \* 15)
* Test case 2: x ≤ 10 (ví dụ: x = 5)
  + Input: 5, Expected: -5
* Test case 3: x = 10 (boundary)
  + Input: 10, Expected: -10

**Câu b:**

Có thể phát hiện lỗi, nhưng không đầy đủ. Để phát hiện lỗi hoàn toàn, cần thêm test-case:

* Test case 4: x ≤ 0 (ví dụ: x = -5)
  + Input: -5
  + Expected output (hàm f1 câu a): -(-5) = 5
  + Actual output (hàm f1 câu b): 2 \* (-5) = -10

Vậycần4 test-case để test đầy đủ hàm f1 ở câu b, bao gồm:

* x > 10 (ví dụ: x = 15)
* 0 < x ≤ 10 (ví dụ: x = 5)
* x = 10 (boundary)
* x ≤ 0 (ví dụ: x = -5)

**Câu c:**

Thực tế hàm này chỉ có 2 nhánh x < 10 và x ≥ 10 vì điều kiện else if (x < 2) không được thực hiện (nếu x < 2 thì x đã < 10, đã được xử lý ở điều kiện đầu tiên).

Số lượng test-case cần thiết: 3 test-case

* Test case 1: x < 10 (ví dụ: x = 5)
  + Input: 5, Expected: 10
* Test case 2: x = 10 (boundary)
  + Input: 10, Expected: 20
* Test case 3: x > 10 (ví dụ: x = 15)
  + Input: 15, Expected: 30

**Câu d:**

Hàm này luôn return 2 \* x bất kể điều kiện gì. Tuy nhiên, cần chú ý: log(x \* x \* cos(x)) có thể gây lỗi runtime nếu x \* x \* cos(x) ≤ 0

Số lượng test-case cần thiết: 2 test-case

* Test case 1: x sao cho x \* x \* cos(x) > 0 (ví dụ: x = 1)
  + Input: 1, Expected: 2
* Test case 2: x sao cho x \* x \* cos(x) ≤ 0 để test lỗi runtime (ví dụ: x = π/2 ≈ 1.57)
  + Input: 1.57, Expected: runtime error

**Câu e:**

Hàm sẽ trả về 0 khi có 2 hoặc 3 số bằng nhau và là số lớn nhất.

Số lượng test-case cần thiết: 5 test-case

* Test case 1: Số khác nhau, num1 lớn nhất
  + Input: (5, 3, 1), Expected: 5, Actual: 5 ✓
* Test case 2: Số khác nhau, num2 lớn nhất
  + Input: (1, 5, 3), Expected: 5, Actual: 5 ✓
* Test case 3: Số khác nhau, num3 lớn nhất
  + Input: (1, 3, 5), Expected: 5, Actual: 5 ✓
* Test case 4: 2 số bằng nhau và lớn nhất
  + Input: (5, 5, 3), Expected: 5, Actual: 0 ✗
  + Input: (3, 5, 5), Expected: 5, Actual: 0 ✗
* Test case 5: 3 số bằng nhau
  + Input: (5, 5, 5), Expected: 5, Actual: 0 ✗

# III. PRATICE 1

## ****Mô tả bài toán:****

**Mục đích:** Giải phương trình bậc 4 trùng phương dạng ax4­ + bx2 + c = 0

**Input:** 3 số thực a, b, c (hệ số của phương trình)

**Output:**

* Trả về số lượng nghiệm thực và lưu các nghiệm vào mảng x[]
* Return -1: Vô số nghiệm (0x² + 0x + 0 = 0)
* Return 0: Vô nghiệm (delta < 0 hoặc 0x² + 0x + c = 0 với c ≠ 0)
* Return 1: 1 nghiệm (phương trình bậc 1 hoặc delta = 0)
* Return 2: 2 nghiệm phân biệt (delta > 0)

## Test Cases:

#include <iostream>

#include <cmath>

using namespace std;

int solveQuartic(double a, double b, double c, double x[]);

int main() {

double x[4];

int result;

// Test 1: Vô số nghiệm (0, 0, 0)

cout << "\nTest 1: a=0, b=0, c=0 (vô số nghiệm)\n";

result = solveQuartic(0, 0, 0, x);

cout << "Kết quả: " << result << " (mong đợi: -1)\n";

cout << (result == -1 ? "PASS" : "FAIL") << endl;

// Test 2: Vô nghiệm (0, 0, 5)

cout << "\nTest 2: a=0, b=0, c=5 (vô nghiệm)\n";

result = solveQuartic(0, 0, 5, x);

cout << "Kết quả: " << result << " (mong đợi: 0)\n";

cout << (result == 0 ? "PASS" : "FAIL") << endl;

// Test 3: Phương trình bậc 1, có nghiệm (0, 2, -8)

cout << "\nTest 3: a=0, b=2, c=-8 (2 nghiệm: ±2)\n";

result = solveQuartic(0, 2, -8, x);

cout << "Kết quả: " << result << " nghiệm (mong đợi: 2)\n";

if (result == 2) {

cout << "Nghiệm: " << x[0] << ", " << x[1] << endl;

cout << "PASS" << endl;

} else {

cout << "FAIL" << endl;

}

// Test 4: Delta < 0 (1, 2, 5)

cout << "\nTest 4: a=1, b=2, c=5 (delta < 0, vô nghiệm)\n";

result = solveQuartic(1, 2, 5, x);

cout << "Kết quả: " << result << " (mong đợi: 0)\n";

cout << (result == 0 ? "PASS" : "FAIL") << endl;

// Test 5: Delta > 0, có nghiệm (1, 0, -4)

cout << "\nTest 5: a=1, b=0, c=-4 (4 nghiệm: ±1, ±2)\n";

result = solveQuartic(1, 0, -4, x);

cout << "Kết quả: " << result << " nghiệm (mong đợi: 4)\n";

if (result > 0) {

cout << "Nghiệm: ";

for (int i = 0; i < result; i++) {

cout << x[i] << " ";

}

cout << endl;

}

cout << (result == 4 ? "PASS" : "FAIL") << endl;

return 0;

}