**EXERCISE 1**

**NAME: NGUYỄN QUỐC HÙNG STUDENT CODE: 3122411060**

1. **VERIFICATION AND VALIDATION**

**Description:** The purpose is to help users to solve a 2-degree equation (a*x*2+b*x*+c).

**Spec:** Given input of *a*, *b*, and *c*; the system returns the outputs of *x*1 and *x*2 (extreme cases are temporarily not considered)

Two systems are developed as follows.

|  |  |  |  |
| --- | --- | --- | --- |
| SYSTEM 1 | SYSTEM 2 | | |
| a b c    Press here to get *x1*  solutions *x2* |  | Step 1: DELTA calculation  a b c    Press here to get DELTA *delta*  (discriminant) |  |
|  | | |
|  | Step 2: Solutions resolving  DELTA *x1*  Press here to get  solutions *x2* |  |
| Code:  x1 = (-b +sqrt(DELTA))/2a x2 = -b -sqrt(DELTA/2a) | Code:  DELTA = (b\*b-4\*a\*c)  x1 = (-b +sqrt(DELTA))/2a x2 = (-b -sqrt(DELTA))/2a | | |

What are the problems of those two systems? Write down your answer here.

TRẢ LỜI: problem của system 1 là:

Công thức x2 bị sai công thức nghiệm. x2 = (-b-sqrt(DELTA))/(2\*a)  
Không tính giá trị DELTA   
Không xử lý ngoại lệ, gồm DELTA < 0 hoặc a = 0

Phép nhân 2a phải là 2\*a thì code mới chạy đúng được

Problem của system 2  
Không xử lý ngoại lệ, gồm DELTA < 0 hoặc a = 0   
UI gây rủi ro làm người dùng dễ làm ra sai sót. VD: người dùng đã nhập a b c, sau đó tính toán 1 phương trình khác và nhập a b c nhưng lần này quên bấm get DELTA, và DELTA cũ vẫn còn trên step2 và từ đó gây sai sót trong tính toán

1. **TEST-CASES**

**Description:** Some input values

* 1. How many test-cases we need for the following function *f*1. What are they?

int f1(int x) { if (x > 10)

return 2 \* x; else

return -x;

}

TRẢ LỜI: chúng ta cần ít nhất 2 test-cases:  
TC1: x>10. ví dụ x=11 thì f1(11)=22  
TC2: x<=10. ví dụ x=10 thì f1(10)=-10

Check if your test-cases can detect error if *f*1 is implemented as follows

int f1(int x) { if (x > 10)

return 2 \* x; else if (x > 0)

return -x; else

return 2 \* x;}

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

TRẢ LỜI: chúng ta cần ít nhất 3 test-cases:  
TC1: x > 10. Ví dụ x = 11 thì f1(11) = 22  
TC2: 0 < x ≤ 10. Ví dụ x = 5 thì f1(5) = -5  
TC3: x ≤ 0. Ví dụ x = -3 thì f1(-3) = -6

* 1. How many test-cases we need to test this function? What are they?

int f2(int x) { if (x < 10)

return 2 \* x; else if (x < 2)

return -x; else

return 2 \* x;

}

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

TRẢ LỜI: chúng ta cần 2 test-cases:  
TC1: x < 10. Ví dụ x = 5 thì f2(5) = 10  
TC2: x ≥ 10. Ví dụ x = 12 thì f2(12) = 24

* 1. How many test-cases we need to test this function? What are they?

int f3(int x) {

if (log(x \* x \* cos(x)) < 3 \* x) return 2 \* x;

else

return 2 \* x;

}

TRẢ LỜI: chúng ta cần ít nhất 2 test-cases:

TC1: log(x \* x \* cos(x)) < 3 \* x. x = 1 thì f3(1) = 2

TC2: log(x \* x \* cos(x)) ≥ 3 \* x. x = 10 thì f3(10) = 20

* 1. Check if your test-cases can detect error if *findMax* is implemented as follows

int findMax(int num1, int num2, int num3) { int max = 0;

if ((num1 > num2) && (num1 > num3)) max = num1;

if ((num2 > num1) && (num2 > num3)) max = num2;

if ((num3 > num1) && (num3 > num2)) max = num3;

return max;

}

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

TRẢ LỜI: chúng ta có 6 test-cases:  
TC1: (1, 2, 3) thì max = 3

TC2: (3, 2, 1) thì max = 3

TC3: (1, 3, 2) thì max = 3

TC4: (2, 1, 3) thì max = 3

TC5: (2, 3, 1) thì max = 3

TC6: (3, 1, 2) thì max = 3

1. **PRATICE 1**

* Mô tả bài toán, các input / output có thể có của bài toán
* Xây dựng các test cases kiểm tra tính đúng đắn chương trình
* Viết đoạn mã tự động kiểm tra chương trình cho bên dưới đúng hay sai?

Input: 3 số thực a b c  
Output: Infinite solutions, No solutions, 2 solutions, 4 solutions, 3 solutions

Các test-cases:  
Trường hợp đặc biệt  
TC1: a=0, b=0, c=0, nghiệm vô số  
TC2: a=0, b=0, c=5, vô nghiệm

TC3: a=0, b=2, c=8, vô nghiệm (y âm)

TC4: a=0, b=2, c=0, 1 nghiệm x=0

TC5: a=0, b=1, c=-4, 2 nghiệm x=±2

Trường hợp a khác 0  
C6: a=1, b=0, c=1 → vô nghiệm (delta<0)

TC7: a=1, b=2, c=1 → nghiệm kép y=-1 (không hợp lệ) → vô nghiệm

TC8: a=1, b=2, c=0 → nghiệm kép y=0 → 1 nghiệm x=0

TC9: a=1, b=-2, c=1 → nghiệm kép y=1 → 2 nghiệm x=±1

TC10: a=1, b=0, c=-1 → 2 nghiệm x=±1

TC11: a=1, b=-5, c=4 → 4 nghiệm x=±1, ±2

TC12: a=1, b=-2, c=0 → nghiệm y1=0, y2=2 → 3 nghiệm x=0, ±√2  
  
Đoạn mã test:  
void test(double a, double b, double c, int expectedCount) {

double x[4];

int n = solveQuartic(a, b, c, x);

cout << "test với a=" << a << " b=" << b << " c=" << c << endl;

if (n == expectedCount) {

cout << "Thành công: (" << n << " solutions)" << endl;

} else {

cout << "Thất bại: expected " << expectedCount << " but got " << n << endl;

}

cout << "----" << endl;

}

#include <iostream> #include <cmath>

using namespace std;

int solveQuartic(double a, double b, double c, double x[]) { if (a == 0 && b == 0 && c == 0) {

return -1;

}

if (a == 0 && b == 0) { return 0;

}

if (a == 0) {

double y = -c / b; if (y < 0) return 0; x[0] = sqrt(y);

x[1] = -sqrt(y); return 2;

}

double delta = b \* b - 4 \* a \* c; if (delta < 0) return 0;

double y1 = (-b + sqrt(delta)) / (2 \* a); double y2 = (-b - sqrt(delta)) / (2 \* a);

int count = 0; if (y1 >= 0) {

x[count++] = sqrt(y1); x[count++] = -sqrt(y1);

}

if (y2 >= 0 && y2 != y1) { x[count++] = sqrt(y2); x[count++] = -sqrt(y2);

}

return count;

}

int main() {

double a, b, c;

cin >> a >> b >> c;

double x[4];

int n = solveQuartic(a, b, c, x);

if (n == -1) {

cout << " Infinite solutions." << endl;

} else if (n == 0) {

cout << "No solution." << endl;

} else {

cout << " The equation has " << n << " real solution(s): "; for (int i = 0; i < n; i++) {

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

}

cout << endl;

}

return 0;

}

**---o0o---**

**(End)**