**EXERCISE 1**

**NAME: TRẦN VĨNH HUY STUDENT CODE: 3122411072**

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:**

* Hệ thống 1:
  + Công thức tính x2 = nhưng thực tế theo code là x2 = -b -sqrt(DELTA/2a) thiếu dấu ngoặc dẫn đến sai công thức.
  + Không có bước tính DELTA
  + Chưa xử lý các trường hợp đặc biệt:
    - DELTA < 0: Phương trình vô nghiệm
    - a = 0: Phương trình bậc nhất
* Hệ thống 2:
  + Chưa xử lý các trường hợp đặc biệt:
    - DELTA < 0: Phương trình vô nghiệm
    - a = 0: Phương trình bậc nhất
  + Sau khi tính được kết quả DELTA ở Step 1, user phải nhập lại DELTA đã tính ở Step 2 dễ dẫn đến sai sót trong nhập dữ liệu và bất tiện.

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:

* Số test case tối thiểu cần: 2
  + TH1: x>10: x = 11, kết quả mong muốn f1 = 22
  + TH2: x<=10: x = 10, kết quả mong muốn f1 = -10
  1. 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:**

* Số test case tối thiểu cần: 3
  + TH1: x>10: Kiểm tra nhánh đầu tiên
    - x = 11, kết quả mong muốn f1 = 22
  + TH2: 0<x<=10
    - x = 5, kết quả mong muốn f1 = -5
  + TH3: x<=0
    - x = -3, kết quả mong muốn f1 = -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:**

* Số test case tối thiểu cần: 3
  + TH1: x<10: Kiểm tra nhánh đầu tiên
    - x = 6, kết quả mong muốn f2 = 12
  + TH2: x<2: Kiểm tra rằng nhánh else if (x < 2) không bao giờ được thực thi do bị nhánh (x<10) bao trùm
    - x = 1, kết quả mong muốn f2 = 2
  + TH3: x>=10
    - x = 12, kết quả mong muốn f2 = 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:

* Số test case tối thiểu cần: 1
* Cả nhánh if và else đều trả về cùng một kết quả: 2 \* x. Do đó điều kiện if (log(x \* x \* cos(x)) < 3 \* x) không ảnh hưởng đến kết quả cuối.
* Về mặt logic kết quả luôn là 2 \* x. Do đó, chỉ cần 1 test case cũng đủ để kiểm thử hàm.
* x = 2 → kết quả mong đợi 4.
  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:**

* Số test case tối thiểu cần: 9
  + TH1: max là num1
    - Input: (5, 3, 1) → mong đợi: 5.
  + TH2: max là num2
    - Input: (2,8, 4) → mong đợi: 8.
  + TH3: max là num3
    - Input: (1, 4, 9) → mong đợi: 9.
  + TH4: num 1 = num 2 > num 3
    - Input: (7, 7, 5) → Output mong đợi: 7 (code trả về 0)
  + TH5: num1 = num3 > num2
    - Input: (6, 2, 6) → Output mong đợi: 6 (code trả về 0)
  + TH6: num2 = num3 > num1
    - Input: (3, 8, 8) → Output mong đợi: 8 (code trả về 0)
  + TH7: Số âm – max là num1
    - Input: (-1, -5, -9) → Output mong đợi: -1
  + TH8: Tất cả âm số
    - Input: (-5, -7, -2) → mong đợi: -2
  + TH9: 3 số bằng nhau
    - Input: (5, 5, 5) → mong đợi: 5 (code trả 0).

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?

#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. (Vô số nghiệm)" << endl;

} else if (n == 0) {

cout << "No solution.(Vô nghiệm)" << 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)**

**Trả lời:**

1. Mô tả bài toán:

* Hàm int solveQuartic(double a, double b, double c, double x[]) thì ta thấy có 3 hệ số a, b, c và một mảng x[].
* Hàm: solveQuartic → Bài toán liên quan đến quartic equation (phương trình bậc 4).

1. Input/Output của bài toán:

* Input: 3 số thực a, b, c (hệ số).
* Output: số nghiệm thực và danh sách nghiệm thực x[i].
  + -1: vô số nghiệm (khi a=b=c=0).
  + 0: vô nghiệm.
  + n: có nghiệm → trả về trong mảng x[].

1. Các test case cần thiết

* Chúng ta cần test các trường hợp:
  + Vô số nghiệm
  + Vô nghiệm
  + 2 nghiệm
  + 4 nghiệm
  + Nghiệm kép
  + Nghiệm đặc biệt x = 0

|  |  |  |
| --- | --- | --- |
| STT | Input (a, b, c) | Kết quả mong muốn |
| 1 | (0, 0, 0) | Vô số nghiệm |
| 2 | (0, 0, 5) | Vô nghiệm |
| 3 | (0, 2, -8) |  |
| 4 | (1, -5, 4) |  |
| 5 | (1, 2, 3) |  |
| 6 | (1, -2, 1) |  |
| 7 | (1, 1, 0) |  |