Problem 243: Stay Away from Me!

Difficulty: Hard

Author: Dr. Francis Manning, Syracuse, New York, United States

Originally Published: Code Quest 2024

Problem Background

It's often important to keep things from colliding with one another; this is true if you're working with self-driving cars, satellites in outer space, or any number of other things. When working on a two-dimensional graph, it's easy to tell if two lines will intersect; if they're not parallel, they'll run into each other eventually. However, we live in a three-dimensional world, and even if you're able to represent the motion of two things as a straight line, checking for intersections – and thus possible collisions – is a bit more difficult.

Problem Description

In a two-dimensional space, a line is usually represented by an equation that defines a starting point on the Y-axis and a slope; for example, $y = 4 + 2x$ represents a line that crosses the Y-axis at (0,4) and has a slope of 2. For this problem, we'll represent lines in three-dimensional space with a set of three parametric equations, like so:

$$x = a + i\alpha$$
$$y = b + j\alpha$$
$$z = c + k\alpha$$

For example, these equations represent two lines:

Line 1	Line 2
$x = 2\alpha$ $y = -1 + 4\alpha$ $z = 6 - 2\alpha$	$x = -5 + 2\beta$ $y = 10 - 2\beta$ $z = -10 + 4\beta$

You'll notice that each set of equations contains the same variable, α (alpha) for line 1, and β (beta) for line 2. While it's not quite the same, these serve a similar purpose to the slope in a two-dimensional equation, in that we can use them to determine if the two lines intersect.

Once we have the equations for two lines, we can combine them to form three equations that can be used to detect if the lines intersect, and if so, where. Using the examples from above:

$$2\alpha = -5 + 2\beta \quad \rightarrow \quad 2\alpha - 2\beta = -5 \ : \ EQ1$$
$$-1 + 4\alpha = 10 - 2\beta \quad \rightarrow \quad 4\alpha + 2\beta = 11 \ : \ EQ2$$
$$6 - 2\alpha = -10 + 4\beta \quad \rightarrow \quad 2\alpha + 4\beta = 16 \ : \ EQ3$$

In each case, we've set the corresponding equations from each line to be equal to each other, then used algebra to get both α and β on the same side. This results in three equations which we can use. Specifically, we can use the equations EQ1 and EQ3 to get a value for β. We've chosen these because the coefficients for α cancel out nicely, but remember that you can use any pair of equations, multiplying both sides of one or both the EQ equations as needed if this isn't the case by default.

$$EQ1 - EQ3$$
$$(2\alpha - 2\beta) - (2\alpha + 4\beta) = -5 - 16$$
$$-6\beta = -21$$
$$\beta = 3.5$$

With this known, we can substitute this value back into any of the EQ equations and determine a value for α; in this example, α is 1. Now we can plug the values for both α and β into EQ2 (the equation we didn't use above) to see if they make sense:

$$4\alpha + 2\beta = 11$$
$$4 * 1 + 2 * 3.5 = 11$$
$$4 + 7 = 11$$
$$11 = 11$$

Excellent! The equation makes sense. This means that the two lines will intersect at some point... but where? You can use the values for either α or β with their respective line's equations to determine that collision point. If your attempt to solve the third equation doesn't work out, this means that the lines won't intersect at any point in space. Whew!

The lines represented in each test case are guaranteed to intersect at no more than one point; put another way, the two sets of equations will represent two distinct lines.

Sample Input

The first line of your program's input, received from the standard input channel, will contain a positive integer representing the number of test cases. Each test case will include two lines containing six numbers separated by commas. These numbers represent the values a, b, c, i, j, and k (respectively) for the equations that represent a line in three-dimensional space.

```
2
0,-1,6,2,4,-2
-5,10,-10,2,-2,4
1,2,3,2,3,4
5,5,5,5,0,5
```

Sample Output

For each test case, your program must determine if the pair of lines defined in the test case intersect.

- If the lines intersect, print a single line containing the x,y,z coordinates of the point of intersection, separating values with commas. Round values to two decimal places and include any trailing zeros.
- If the lines do not intersect, print a single line containing the word "Miss"

```
2.00,3.00,4.00
Miss
```