Tess L. Ation ran into a little problem last week when she demonstrated the beta version of her new drawing software. On the screen she had an elegant demonstration design that illustrated every feature of her program; it had taken her hours to produce it. She was just putting the finishing touches on it as a group of potential investors entered the room to see the demonstration.

The presentation went well. Near the end, Tess clicked on a control panel button and told her audience, "This is the 'snap to grid' control. It forces control points, such as vertices, to jump to the nearest grid point. Here, let me show you," and she placed three bright red dots on the screen. Each one appeared at the grid point nearest to where she clicked. ("Luckily all control points in my demo design were already at integer coordinates. But I will have to remember to delete these three red dots before I save my diagram," she thought to herself.) "Now I'll step into the next room and get out of your way so you can discuss the system among yourselves and get a closer look at the screen, but please don't touch anything, since I haven't saved that file yet."

A few minutes later, the group joined Tess. One of the visitors stepped up to Tess and said, "I hope you don't mind, but I wanted to try it myself. Don't worry, I just played with the x-scale and y-scale controls a little bit." The next person said, "Sorry if this is a problem, but I really wanted to get a feel for the speed of display, so I just played around with the translation tool." And a third person said, "I couldn't resist just one tiny test: I rotated the image just so I could see all of the vertices snap to the nearest grid points after the rotation."

The person who played with the rotation tool remembered going first, but the other two could not recall their order. The three remembered only a few details of the changes. The x- and y-scaling factors had been (possibly negative) nonzero integers; the center of scaling was the origin (0, 0). The x- and y-translation amounts had been integers. Rotation had been specified by a point with integer coordinates (x, y) on the perimeter of a square of width 20 centered at the origin (hence, $-10 \leq x, y \leq 10$ and the absolute value of x or y or both was 10). The tool rotated the drawing around the origin such that the positive x-axis would pass through (x, y) afterwards. Snapping took place after this rotation (coordinates with a fractional part of 0.5 were rounded away from zero).

After they left, Tess looked at her design — it was completely changed! She had not yet implemented the "undo" feature, and she had not saved the diagram prior to giving the demonstration. However, the three identical red dots were still there (transformed to other integer grid locations, of course), and Tess could remember the integer coordinates where she had originally placed them. Obviously, someone else might have altered the drawing without saying anything to her, but she could write a program to see if it was possible to reconstruct the sequence of alterations. Can you too?

Input

The input contains several test cases. Each test case consists of six pairs of integers x_i and y_i ($-500 \leq x_i, y_i \leq 500$ for $1 \leq i \leq 6$), three pairs per input line. The first three pairs represent the distinct initial locations of the three red dots. The last three pairs represent the distinct final locations of the three dots. The indexing of the pairs in each group of three is not significant: for example, (x_1, y_1) could have been mapped to any of (x_4, y_4), (x_5, y_5) or (x_6, y_6).

The last test case is followed by a line with six zeros.

Output

For each test case, display its case number followed by one of the following three messages:

- '`equivalent solutions`' to indicate that there are one or more valid transformations, and of them have the same effect on the whole drawing (no matter what the whole drawing looks like).
- '`inconsistent solutions`' to indicate that there are several valid transformations, but in general not all of them map the entire drawing in the same way (some drawing is mapped differently by two valid transformations).
- '`no solution`' to indicate that neither of the first two cases occurs.

A valid transformation is a combination of rotation, translation and scaling (or rotation, scaling and translation) which satisfies the restrictions described above and maps the initial set of red dots to the final set (occupying all three final locations).

Follow the format of the sample output.

Sample Input

```
3 0 4 0 1 4
-2 -4 -1 3 3 -4
0 1 1 1 2 1
1 2 2 2 3 2
1 0 2 0 3 0
3 3 1 1 2 2
1 0 2 0 3 0
3 2 1 1 2 2
2 3 0 6 1 2
2 3 0 6 1 2
0 0 0 0 0 0
```

Sample Output

```
Case 1: equivalent solutions
Case 2: inconsistent solutions
Case 3: no solution
Case 4: inconsistent solutions
Case 5: equivalent solutions
```