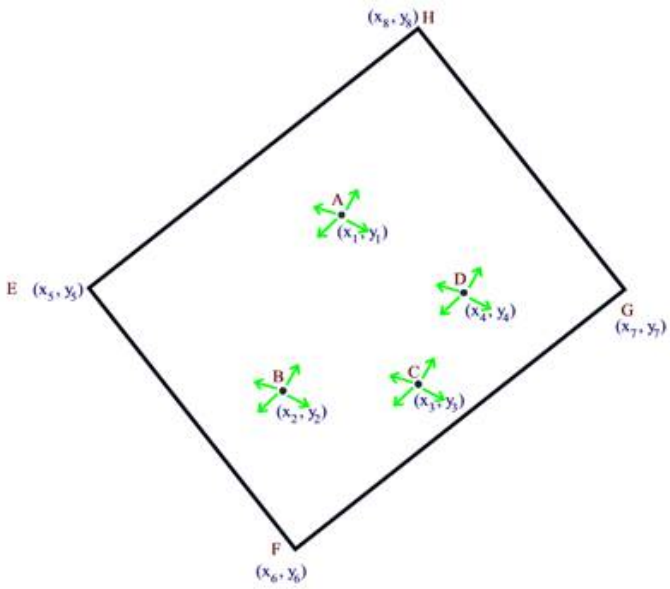
You have to help a group courageous people to find the lost treasure of your country, like the movie "**National Treasure - Book of Secrets.**" To discover the national treasure that courageous group has to pass through many traps, solve many riddles and so on.

Like the film there are four people in the group and to reach the treasure they have to jump on a plate that is placed on a pillar with sharp edge at the top. Before any of them jumps on the plate, it is in a stable position. For simplicity you can assume that the plate is rectangular in shape, has equal thickness at all places, it is solid and made of same material. Four people of equal weight jump on the plate at the same time but they can jump only at some certain points of the plate. So they cannot always jump in such a way that the plate is stable after they jump on it. Therefore after jumping on the plate they must run and change their position to make the plate stable again.

A plane of equal thickness is placed on a pillar and four persons jump on it.

They run to change their position and keep the plane balanced.



After jumping on the rectangular plate the four people can run at any direction they want.

Given the shape of the plate (always rectangular) and location of the four people just after jumping on it, your job is to find the minimum distance they have to cover in total to make the plate stable again. After jumping the four people can run towards any direction. You don't need to print the minimum distance, but you need to print the final position of the four people after covering the minimum distance in total.

Input

The input file contains around 15000 sets of inputs. Each set is described with 16 floating-point numbers scattered in two lines. The first line contains eight floating-point numbers x_1, y_1, x_2, y_2, x_3, y_3, x_4 and y_4 ($-10 \leq x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4 \leq 1010$) that denotes the coordinates of the four people just after jumping on the plate. So the coordinate of the four people are (x_1, y_1), (x_2, y_2), (x_3, y_3) and (x_4, y_4) respectively. The second line contains another eight floating-point numbers x_5, y_5, x_6, y_6, x_7, y_7, x_8, y_8 ($-10 \leq x-5, y_5, x_6, y_6, x_7, y_7, x_8, y_8 \leq 1010$). These eight floating-point numbers actually denote the coordinates of the four corners of the plate in counter clockwise order. The shape of the plate is always rectangular. You can assume that the jumping locations (x_1, y_1), (x_2, y_2), (x_3, y_3) and (x_4, y_4) are strictly within the plate boundary and all four people have equal weight. Input is terminated by two lines, each containing eight zeroes. All floating-point numbers in the input has twelve digits after the decimal point. The coordinates are all in two dimensions so they actually are given to specify the size and orientation of the plate and the location of the people with respect to the plate. So if a person stands still on the plate and the plate moves up and down or rotates, the coordinates of the person or coordinate of four corners of the plate does not change. In the sample input it is shown that each line contains four floating-point numbers. But it is only for lack of horizontal space on paper. In the judge input file there are eight floating-point numbers in each line.

Output

For each set of input produce four lines of output. Each line contains two floating-point numbers that denote the final position of one of the four persons. The final position of the four persons should be mentioned according to the same order given in the input and should make the plate stable. Also the final position of all four persons must be inside (or on the boundary of) the plate as well and ensure that the total movement made by the four persons is minimum. If there is more than one solution, any one will do. All floating-point numbers in the output should contain 12 digits after the decimal point. Print a blank line after output for each set. There is an special judge for this program, so small precision error will be ignored.

Sample Input

```
731.637000000000 437.595000000000 296.162000000000 402.836000000000
493.625000000000 260.917000000000 526.376000000000 237.611000000000
631.933553096878 841.348627667446 158.076619219714 651.913864882162
360.066446903122 146.651372332554 833.923380780286 336.086135117837
0.000000000000 0.000000000000 0.000000000000 0.000000000000
0.000000000000 0.000000000000 0.000000000000 0.000000000000
0.000000000000 0.000000000000 0.000000000000 0.000000000000
0.000000000000 0.000000000000 0.000000000000 0.000000000000
```

Sample Output

```
715.687000000000 596.855250000000
280.212000000000 562.096250000000
477.675000000000 420.177250000000
510.426000000000 396.871250000000
```