You live in a flat world and you have to carry some goods to three destinations A, B, C from a storeroom. You know the location of A, B and C and you have to find an optimal location G for the storeroom and build the storeroom at G.

But for carrying goods you have only one truck available and that can drive through any place/location you want. The truck will initially be located at G. But this truck is not large enough to carry goods for more than one place at a time. So for minimum path covering what you do is:

1. Always drive from one place to another in straight line.
2. Load goods in the truck at G.
3. Carry these goods to the nearest destination to G.
4. Unload the goods at the nearest destination.
5. Drive the empty truck back to G.
6. Load good in the truck at G.
7. Carry these goods to the 2nd nearest destination from G.
8. Unload the goods at the 2nd nearest destination.
9. Drive the empty truck back to G.
10. Load goods in the truck at G.
11. Carry these goods to the farthest destination from G. And of course stay at the farthest destination, as you have to carry nothing else.

If you had known the location of G then to find the minimum driving length would have been very easy. But for this problem your job is to find a location of G for which the total path length would be minimum and report this minimum driving length.

Input

The input file contains less than 11000 lines of input.

Each line contains six integer numbers A_x, A_y, B_x, B_y, C_x, C_y. You can assume that $(0 \leq A_x, A_y, B_x, B_y, C_x, C_y \leq 1000)$. These integers denote that the location of A, B and C in two-dimensional Cartesian coordinate system is (A_x, A_y), (B_x, B_y) and (C_x, C_y) respectively.

A line containing six negative numbers terminates the input.

Output

For each line of input except the last one produce one line of output. This line contains the serial of output followed by a floating-point number d, which denotes the minimum driving length needed from the optimal location of G. This number should have eight digits after the decimal point. Errors less than 10^{-7} will be ignored. Look at the output for sample input for details.

Sample Input

```
0 0 15 0 8 1
-1 -1 -1 -1 -1 -1
```

Sample Output

```
Case 1: 22.20439337
```