11704 Caper pizza

Brunno Doiuna is very fond of caper pizzas, which he always likes to share with his girlfriend. As she also loves capers, it is of the utmost importance, in order to avoid unnecessary quarrels, to split the pizza into two equally-sized slices in such a way that each half contains exactly the same number of capers. However, most caper pizzas also contain a number of peppercorns, and neither Brunno nor his girlfriend likes them. Therefore, it is also crucial that each of the two halves contain the same number of peppercorns. As you can easily observe, depending on the position of capers and peppercorns on the pizza, it is not always possible to make a straight-line cut into the pizza in such a way that the two slices have the same area and the same number of capers and peppercorns lie in each resulting piece. Your task is to design a program to decide if it is possible to make such a cut or not, knowing the positions of capers and peppercorns.

Input

We will assume that the pizza is circular and is centered at the origin, and is big enough to contain all capers and peppercorns. We also assume that there is an even number of capers and an even number of peppercorns, and that no cut goes through any of the capers or pepper balls. Additionally, no pair of peppercorns, capers, or a peppercorn and a caper are aligned with the origin, or form an angle of less than 10^{-6} degrees with the origin.

There can be multiple test cases, one after the other. The first line of a test case contains two even integers $c \geq 0$ and $p \geq 0$ (where $2 \leq c + p \leq 30000$) separated by a space, the number of capers and peppercorns, respectively. Each of the following c lines describes the position of a caper using two floating point numbers, separated by a space, representing its x and y coordinates. Each of the next p lines holds two floating point numbers, the x and y coordinates of a peppercorn. A blank line follows each test case.

The last line of input will contain '`-1 -1`'. This marks the end of input — do not write any output for this special last line.

Output

'`YES`' for a positive answer, '`NO`' otherwise.

Sample Input

```
2 2
1 1
1 0
0 1
-1 1

2 2
1 1
-1 1
0 1
0.1 -1

-1 -1
```

Sample Output

```
NO
YES
```