FoodSberry

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

With the speed of light are food technologies conquering cities and towns of Berland, and, especially, its gorgeous capital City N.

FoodBerry is the largest food delivery that operates in City N. Consider the city as a two-dimensional square with sides parallel to coordinate axis. The bottom left corner of this square has coordinates $(-10^4, -10^4)$ and the top right corner has coordinates $(10^4, 10^4)$. All coordinates are given in meters.

FoodBerry has one Super Large Warehouse located at $(0, 0)$ and n dark stores (smaller warehouses), the i-th of them is located at point (x_i, y_i). Each dark store has two types of delivery: by foot and by car. Delivery by foot can be used for all points that are located no more than a meters from this dark store, while delivery by car can be used for up to b meters distances ($a \le b$, obviously). During one day each dark store is able to execute no more than c deliveries. Moreover, due to the limitation of vehicles, no more than d of these deliveries can be done by car ($d \le c$).

Super Large Warehouse is located at $(0, 0)$ and is capable of performing any number of deliveries to any locations in the city. However, using this option is expensive and FoodBerry tries to avoid it to for as long as possible.

In this problem we are going to analyze one day of FoodBerry in City N. There were m orders during that day, they are given in the order they were made and processed. The i-th order asks for the delivery to point (x'_i, y'_i). For the purpose of this problem we introduce some bold assumptions.

1. First all orders were received. Then all deliveries were executed.
2. No two orders appear in the same moment of time.
3. After each order appears, there is enough time to re-process everything. Scheduling center assumes there will be no more orders today and makes a distribution of orders between dark stores (and the warehouse). Moreover, for each delivery it is determined whether it should be done by foot or by car. This distribution is completely re-done after any new order appears, it doesn't need to depend on the previous distribution in any way.

Of course, each distribution is done with respect to limitations on distance and quantity of orders per dark store given by parameters a, b, c and d. If there is a distribution of orders that satisfies all the requirements and executes all the deliveries without using Super Large Warehouse, the scheduling program goes for it.

Assuming all distributions are made optimal, what is the minimum order index i such that there will be no way to make a valid distribution without using Super Large Warehouse?

Input

The first line of the input contains six integers n, m, a, b, c and d ($1 \le n, m \le 500$, $1 \le a \le b \le 30\,000$, $1 \le d \le c \le 1000$). Here n is the number of dark stores in City N, m is the number of orders to process, a is the maximum possible distance for delivery from dark store by foot, b is the maximum possible distance for delivery from dark store by car, c is the maximum possible total number of deliveries a dark store can execute a day, and d is the maximum possible number of deliveries by car a dark store can execute a day.

Then follow n lines containing two integers x_i and y_i ($-10\,000 \le x_i, y_i \le 10\,000$) each — coordinates of dark stores.

Finally go m lines with coordinates of orders. Each of them contains two integers x'_i and y'_i ($-10\,000 \le x'_i, y'_i \le 10\,000$).

Any points of the input can coincide, i.e. there can be two or more dark stores in the same point, two or more orders in the same point, a dark store and an order in the same point, a dark store or an order at point $(0, 0)$ and so on.

It is possible that some order is located out of reach of any dark store. Note that it is still possible to execute all the orders as FoodSberry has Super Large Warehous that is capable of doing any number of deliveries to any location in City N.

Output

If it is possible to obey all the requirements and satisfy all m orders without using Super Large Warehous, print `-1` in the only line of the output.

Otherwise, print minimum possible index i such that Super Large Warehouse is required to satisfy first i orders.

Examples

standard input	standard output
1 3 1 3 2 1 1 1 2 1 2 2 1 2	3
3 6 1 1 2 2 0 1 -2 1 2 1 -1 1 1 1 0 2 0 0 -2 1 2 1	-1