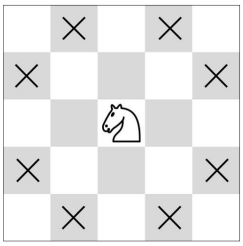
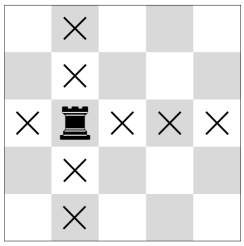
Problem K
Lone Knight
Time Limit: 7 Second(s)

In the game of chess, a knight moves as shown in the picture below; each move is one square horizontally and two squares vertically or two squares horizontally and one square vertically. A rook can move any number of squares horizontally or vertically, but not both in the same move. If a square can be reached by a rook in one move, that square is said to be attacked by the rook.





Consider an infinite chess board, with squares that can be indexed by integer coordinates. There is a white knight on the board on a square, and it wants to go to another square. However, there are also a number of black rooks on the board. The knight can make as many moves as it needs to get to its target square, but it cannot stop on a square that is attacked by or occupied by a rook. The rooks don't move.

Can the white knight reach its target square? You are to answer that question many times!

Input

The first line of input contains two integers n and q ($1 \leq n, q \leq 1{,}000$), where n is the number of black rooks and q is the number of queries.

Each of the next n lines contains two integers x and y. This indicates that there is a black rook at (x, y). No two rooks share the same square.

Each of the next q lines contains four integers x_s, y_s, x_t and y_t. This is a query, where the white knight starts at square (x_s, y_s) and wants to move to square (x_t, y_t).

All square coordinates in the input are no larger than 10^9 in absolute value. It is guaranteed that in every query the knight's initial and target squares are not attacked by or occupied by any rook, and the target square is not the same as the initial square.

Output

For each query, output on a single line `1` if the knight can reach the target square, or `0` otherwise.

Sample Input 1

```
6 6
10 14
1 0
0 1
4 9
9 13
5 9
2 2 3 4
2 2 2 4
2 2 6 4
2 2 2 10
7 11 6 2
6 2 8 12
```

Sample Output 1

```
1
0
0
1
0
1
```

Sample Input 2

```
8 10
0 0
1 1
5 5
8 8
11 11
14 14
18 18
19 19
17 10 13 9
15 15 15 9
7 3 17 4
15 15 12 3
9 17 3 3
4 4 9 4
12 12 2 6
10 15 6 6
15 17 4 16
-1000000000 -999999999 15 7
```

Sample Output 2

```
1
0
0
0
0
0
0
0
0
0
```