Problem 4: communication towers

In a communication network, a number of towers are placed strategically for transmission purposes. Each of these towers has an effective range (or radius covered) of, say, 5 km. In order to have good reception of the signal, and to ensure one would not be disconnected in case of failure of one of the towers, it is recommended to be in a location covered by at least two towers. Your job is to check that each of the locations provided are indeed covered by at least two towers. Note: in order to be covered by a tower, the direct distance between the location and the tower must be smaller than 5. As a reminder, here is the formula to calculate the distance between two points (one at coordinates <x1,y1> and one at coordinates <x2, y2>):

$$\sqrt{(x1-x2)^2+(y1-y2)^2}$$

INPUT: The input begins with a single line containing N, a positive whole number that indicates the number of towers. Then there are N lines. Each consists of the coordinates of each of those towers, as 2 positive whole numbers separated by a space: the X coordinate, then the Y coordinate. This is followed by an empty line, and then a positive whole number M (on a single line) indicating the number of locations to be checked. There are then M lines. Each consists of the coordinates of each of these locations (again, as 2 positive whole numbers separated by a space: X coordinate, then Y coordinate).

OUTPUT: The index of the locations that are not properly covered (the first location in the input file having an index of 1, the second one having an index of 2, and so on). You can assume that there will always be at least one location not properly covered. The indexes should be displayed in ascending order.

For the example below, the first location is properly covered, as it is clearly within a distance of 5 of both the towers 1 and 3. However, the second location is not properly covered because it is within the range of tower 3 only. The third location is way out of range of any of the towers.

EXAMPLE INPUT:

```
3
5  3
2  8
6  9

3
5  6
8  11
30  9
```

EXAMPLE OUTPUT:

```
2
3
```