25-th BULGARIAN OLYMPIAD IN INFORMATICS

National Round, 1. - 3. 05. 2009

Task A1. MINESWEEPER

As any modern teenage student, Elly is involved in many extracurricular activities – theatrical art, cliff climbing, debates and mine sweeping. The last one is probably a bit shocking for you, but the job is quite interesting and not so complicated. As a beginner mines sweeper Elly has to find only where the mines are positioned in the mine field and surround them with a warning band (in order to prevent pedestrians from walking upon them). Elleonor from the other hand has no troubles at all finding the mines – her modern mobile phone ("phone" is probably a bit insulting for the device she possesses) has a 12-megapixel camera, Wi-Fi, GPS, and metal detector, which makes the job a piece of cake. After making a photo of the mine field she has to surround all mines with one piece of warning band, trying to minimize its length. Each mine is a circle with a radius R and coordinates of the center (x, y). Elly has showed you the map of the mine field and wonders how long should the band be. Can you help her and to write a program **`msweeper`** to determine the length of the shortest band, which can surround the mines, given the coordinates and the radius of the mines?

Input

On the first line of the standard input two integer numbers N and R will be given – the number of the mines and their radius. On each of the next N lines the coordinates x and y of the center of a mine will be given. The mines can overlap (in order to create more powerful blast, for example).

Output

On a singe line of the standard output the program has to print the minimal length of a band. Your answer will be considered correct if it differs from the author's less than 0.001.

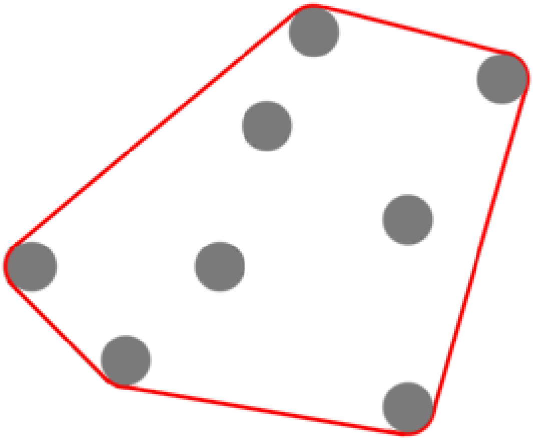
Constrains: N – integer, $0 < N < 10{,}001$; R – integer, $0 < R < 42$;
x and y – integer numbers, $-20\,000 < x, y < 20\,000$.

EXAMPLE

Input

8 1
1 4
3 2
7 9
5 4
9 5
6 7
9 1
11 8

Output
34.408



25-th BULGARIAN OLYMPIAD IN INFORMATICS
National Round, 1. - 3. 05. 2009

Task A2. MONEY-BOXES

The programmer Pesho is a very thrifty person. The money which he earn from web-sites design he spares in *N* money-boxes ($1 \leq N \leq 100000$), labeled by the integers from 1 to *N*. The intention of Pesho is to buy a new super-computer. In order to avoid the temptation to spend the money for rubbish, before the necessary sum is collected, he dropped all keys of the money-boxes in random way inside the money-boxes themselves. Fortunately, Pesho marked on a list of paper the key of which money-box in which money-box was dropped. Nowadays the necessary money was finally collected and Pesho has to open all money-boxes so as to take the money. Because no keys are available one or more money-boxes have to be broken. And Pesho don't like to break anything. He realized that when a broken money-box contains a key of another money-box then the second money-box could be unlocked and no need to be broken. Anyway, the task of minimizing the number of the broken money-boxes seems not easy. Help Pesho. Write a program named `boxes` to determine the minimal number of money-boxes that have to be broken in order to collect all spared money.

Input
The program has to solve two test cases for one run. Each test case starts with a line of the standard input containing the number *N* of money-boxes. Then *N* lines follow with one integer from 1 to *N* – on *i*-th of these lines is given the label of the money-box, where the key of the money-box with label *i* is dropped.

Output
On a single line of the standard output the program has to print the minimal number of money-boxes that have to be broken for each of the two test cases, in the order they are given in the input. The two numbers have to be separated by an interval.

EXAMPLE

Input	Output
4 2 1 2 4 3 3 3 3	2 1

Explanation of the example: In the first test case the money-box 4 obviously has to be broken, because it contains its own key. To open the other money-boxes will be enough to break, for example, the money-box 1 – it contains the key that will open money-box 2, and money-box 2 contains the key that will open money-box 3. In the second test case will be enough to break the money-box 3.

25-th BULGARIAN OLYMPIAD IN INFORMATICS

National Round, 1. - 3. 05. 2009

Task A3. TITANIC

The awful events from 1912 can happen again! With this touthg the young mathematician Drown Waterswallower for a second week dreams that he is under the hull of a ship, turned upside-down and already filled with water. With the air inside his lungs he will not live for long, so every second is precious for him. Drown Waterswallower has mentally divided the rectangular ceiling (i.e. the surface above him) to *N x M* square areas. Because of the heavy damage the ceiling is very uneven – no two areas are on the same height. With a waterproof torch our hero can light up any square of the ceiling and to determine its height. Drown has some knowledge about physics, so he knows that air could be left only under a square, which is higher than all of its neighbor squares (two squares are neighbors if they have a common side). The squares on the ceiling border have less than four neighbors. Give the poor mathematician a chance to live longer by writing a program named **`titanic`**, which conducts a series of checks about the height of certain pieces on the ceiling and finds a square such that all of its neighbors are lower. The program has to print a question on the standard output and will receive immediately the answer from the standard input. The program could ask no more than *K* questions.

Constraints: $1 \le N, M \le 1000$; $K = 3050$.

Communication:

- in the beginning on the standard input *N* and *M* are given, separated by a space;
- printing on the standard output a line **`0`** *x y* ($0 \le x < N$, $0 \le y < M$), the program will be able to read from the standard input the height of the square with coordinates *x* and *y* (a 32-bits signed integer);
- finding a local maximum in the square with coordinates *x* and *y* the program has to print on a single line of the standard output **`1`** *x y* and to stop.

Example dialog:

```
» 3 2
« 0 0 0
» 3
« 0 1 0
» 4
« 0 2 0
» 1
« 0 0 1
» 7
« 0 1 1
» 6
« 0 2 1
» 2
« 1 0 1
```

Grading: If the program works more than the specified time limit, doesn't give an answer (a line starting with **`1`** followed by two coordinates), asks more than *K* questions about heights, gives wrong answer for at least one of the combined tests, or creates run-time problems then you will receive 0 points for the corresponding combination of tests. Otherwise you will receive the full score for tha tests.