Crop Circles

Alien tourists love visiting Earth to see human artists draw crop circles in fields. You are in charge of producing the most intricate layout of crop circles yet! The field you are working in can be thought of as an infinite 2D plane. Centuries of research on xenoaesthetics gives some basic restrictions you must follow: You must draw exactly N circles, the i-th of which must be centered at the integer coordinates (x_i, y_i).

Your job is to select a non-negative radius r_i for each circle so that the circles do not overlap. They may however touch at their edges. Note that the radius you select does not have to be an integer. Formally, circles i and j overlap if and only if:

$$(x_i - x_j)^2 + (y_i - y_j)^2 < (r_i + r_j)^2$$

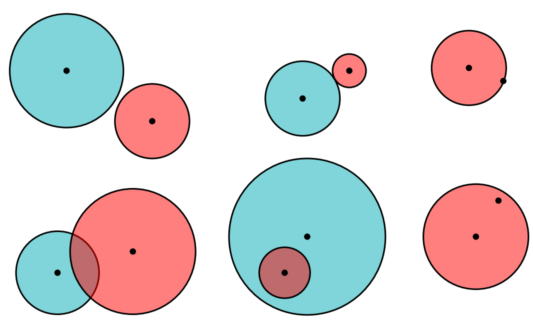


Figure 1: The three examples on the top show non-overlapping circles. The three examples on the bottom show overlapping circles. Note that the two examples on the right feature a circle with radius 0.

The *beauty* of your layout is the sum of the **circumferences** of your circles. You do not have to produce the maximum total beauty possible, instead you are scored on the total beauty you are able to achieve. Please read the Scoring section below.

Subtasks and Constraints

For all subtasks, you are guaranteed that:

- $1 \leq x_i, y_i \leq 1\,000\,000\,000$ for all i.
- No two circles share the same center. That is, $(x_i, y_i) \neq (x_j, y_j)$ for all $i \neq j$.

In all test cases (other than the sample case), the values of x_i and y_i are chosen uniformly at random subject to the constraints above.

Additional constraints for each subtask are given below. Each subtask has exactly 5 test cases.

Subtask	Points	N
1	10	10
2	10	20
3	10	50
4	10	100
5	15	200
6	15	500
7	15	1000
8	15	2000

Input

- The first line of input contains N.
- The following N lines describe the circle centers. The i-th line contains x_i and y_i.

Output

Output N lines: the i-th line should contain r_i, the radius of the i-th circle.

Scoring

If any two circles overlap, or if you give a radius less than 0, then you will score 0%.

Otherwise, let OPT be the maximum beauty possible for the test case, and SOL be the beauty your solution achieves. If $SOL = OPT$, you will score 100%.

Otherwise, you will score $-20 \times log_{10}(1 - \frac{SOL}{OPT})\%$ for the test case (up to a maximum of 100%). In particular:

SOL/OPT ratio	points (%)
0.5	6.02
0.6	7.96
0.7	10.46
0.8	13.98
0.9	20
0.99	40
0.999	60
0.9999	80
0.99999	100

To ensure that your output contains sufficient precision, you should use the 'double' data type in C++ and output the radius of each circle to at least 9 digits of precision.

To output a variable defined as `double x;` to standard output with printf/scanf use `printf("%.9f");`

To output to cin/cout, first `#include <iomanip>`. Then, inside your main function before any other `cout` statements, write:

```
std::cout << std::fixed << std::setprecision(9);
```

Sample Input

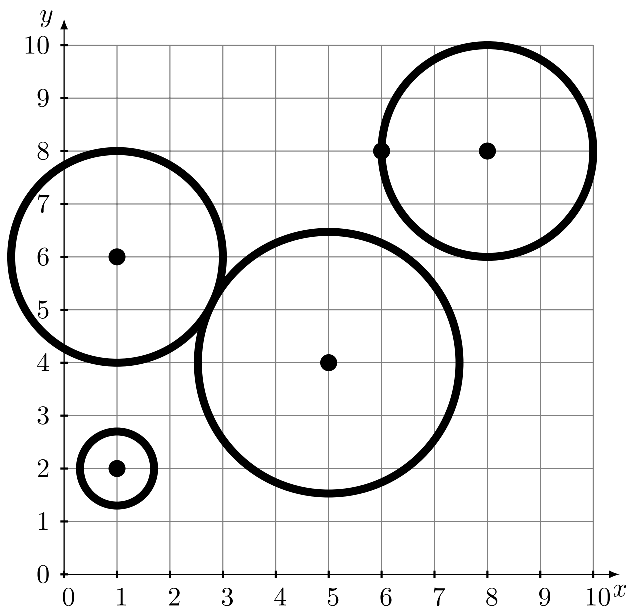
```
5
1 6
5 4
1 2
8 8
6 8
```

Sample Output

```
2.000000000
2.472135955
0.700000000
2.000000000
0.000000000
```

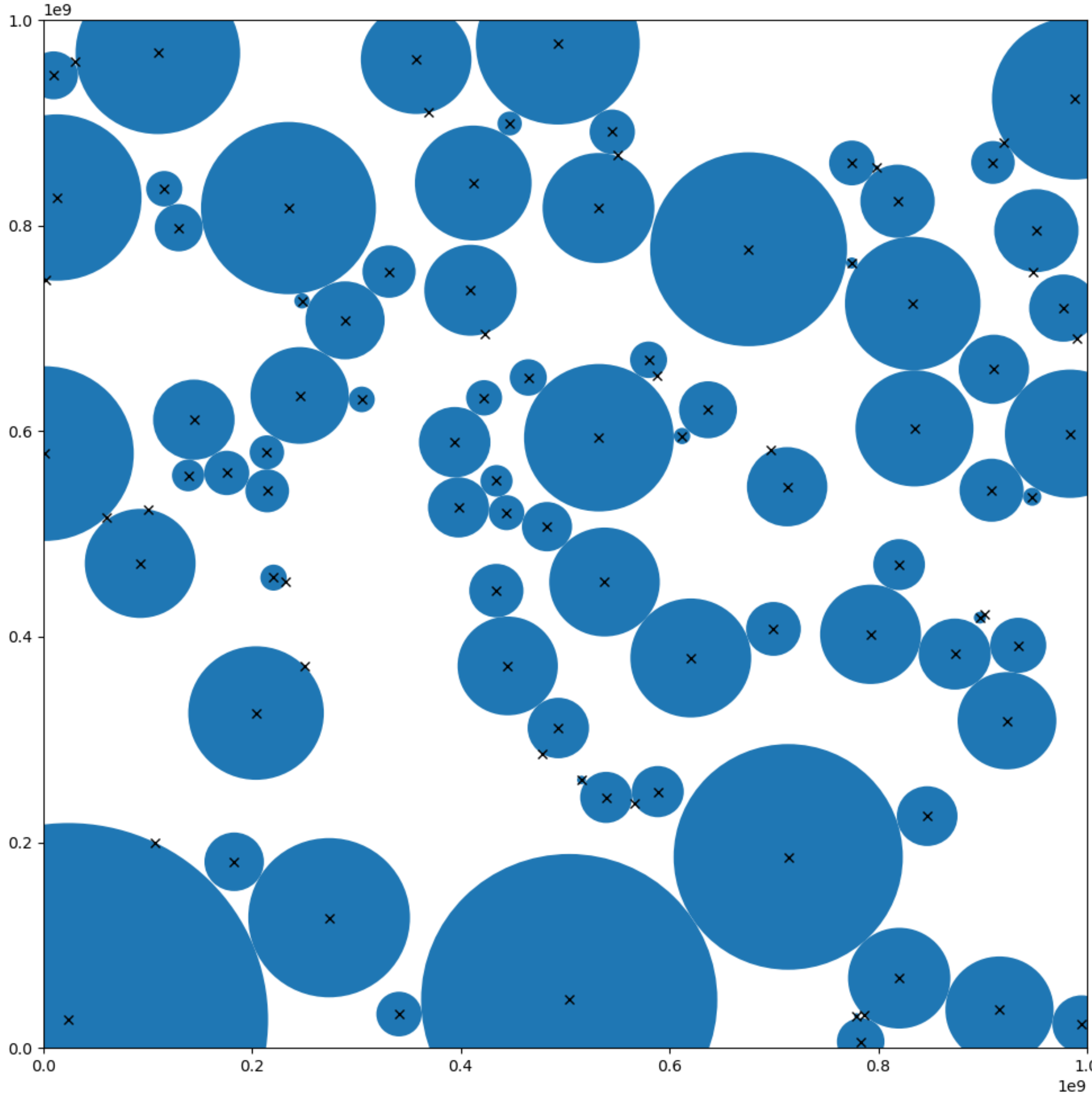
Explanation

The total beauty of the sample output is 45.064. . . , while the maximum beauty possible for the sample input is 53.232. . . , so this output would score 16.28% of points.



Sample output

One optimal solution for a randomly generated input with $N = 100$ is shown below.


1e9
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1e9