

Lines

After reading on Facebook that Claude Code has ended programming, Fuki has decided to switch to doing physics.

He thinks that he has discovered a new physics formula for determining how many kilograms of burek he will eat, given the time since his last meal. He thinks that the formula is of the form $y = m \cdot x + b$, where y is the amount of burek he will eat and x is the time since his last meal. m and b are some constants representing the slope and y-offset respectively.

He has collected n datapoints to verify his breakthrough. The datapoints are given by (x_i, y_i, d_i) , where x_i is the time since his last meal, y_i is the scale's measurement of the amount of burek eaten, and d_i is the inaccuracy of the scale with which he measured the burek. Therefore, the actual amount of burek he ate is in the range $[y_i - d_i, y_i + d_i]$, and he does not know the exact weight. For the sake of the task m can be negative and y_i can be smaller than d_i .

We can represent the i -th datapoint as a vertical line segment whose endpoints are $(x_i, y_i - d_i)$ and $(x_i, y_i + d_i)$.

Fuki now wants to know whether there exists a non-vertical line that passes through all of these segments. In other words, the line must intersect every segment.

If such a line does not exist, report that it does not. Otherwise, report the minimal and maximal possible slope of such a line.

Note that the lines with minimal and maximal slope do not need to have the same y-offset.

Input

In the first line is a positive integer n .

Then follow n lines with one integer x_i and two decimal numbers y_i and d_i . The numbers have at most 9 digits after the decimal point.

It is guaranteed that all x_i are unique.

Output

If no slope exists, output -1 . Otherwise, output 2 real numbers: the minimal and maximal possible slope.

For both numbers an absolute or relative error of up to 10^{-6} is allowed. Formally, if your output is x' and the expected output is x , then your output is correct if $\frac{|x-x'|}{\max(1,|x|)} \leq 10^{-6}$.

Implementation tips

For more precise calculations use the C++ `long double` type. Also, when printing floating point types make sure to specify precision. Below is a code snippet of how one might do that.

```
long double val = 10;
std::cout << std::fixed << std::setprecision(9) << val << "\n";
```

Make sure to include the `<iomanip>` header when using `setprecision`.

Constraints

It is guaranteed that if a valid line exists, then the maximal slope minus the minimal slope is at least 0.1. If no valid line exists, then no valid line exists even when every d_i is increased by 10^{-3} . It is also guaranteed that for any valid m , $|m| \leq 10^3$.

- $2 \leq n \leq 2 \cdot 10^5$
- $0 \leq x_i \leq 10^6$
- $0 \leq y_i \leq 10^9$
- $0 < d_i \leq 10^9$

Subtasks

Subtask 1 (5 points)

$$n = 2$$

Subtask 2 (7 points)

It is guaranteed that there is a line with slope 1 that passes through all segments.

Subtask 3 (19 points)

$$n \leq 100$$

Subtask 4 (29 points)

$$n \leq 2000$$

Subtask 5 (40 points)

No additional constraints.

Examples

Input 1

```
2
0 0 1
10 10 1
```

Output 1

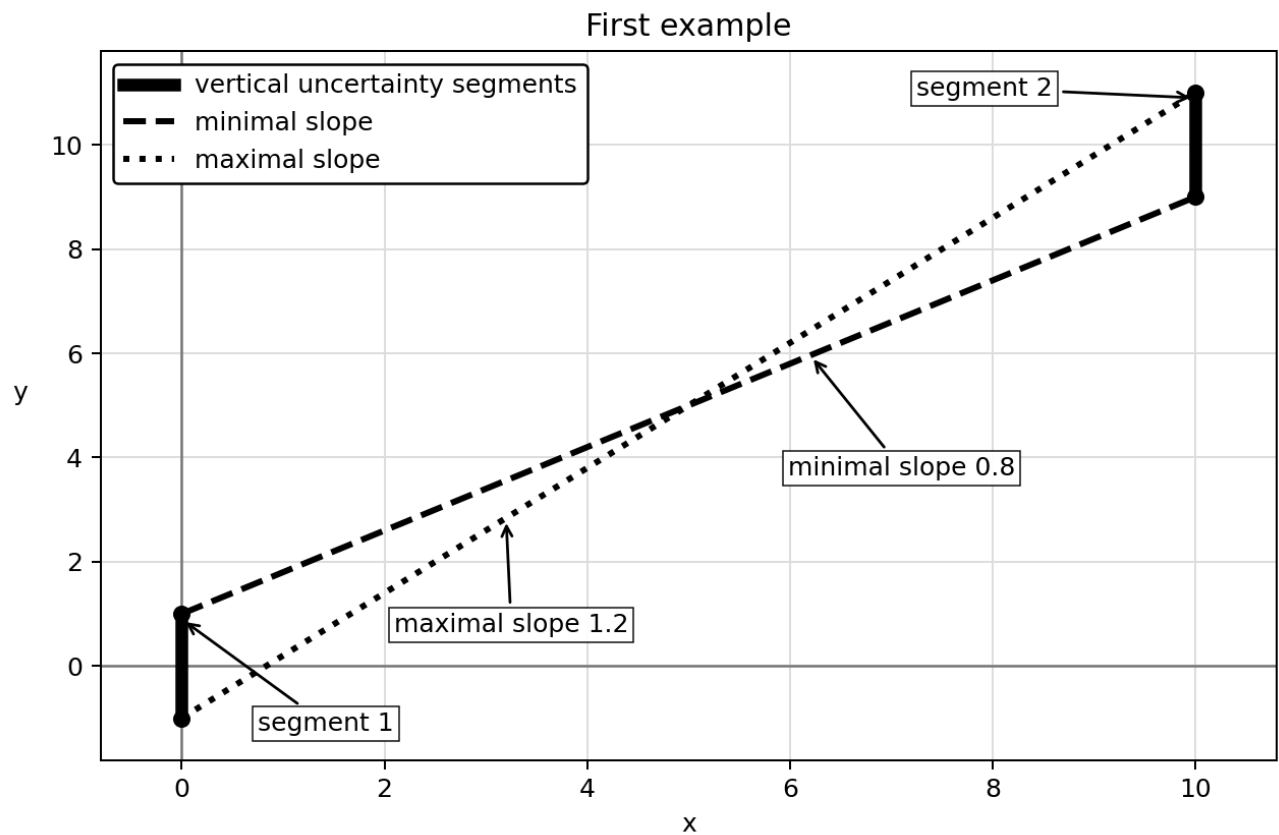
```
0.800000000000 1.200000000000
```

Explanation 1

The first segment goes from $(0, -1)$ to $(0, 1)$, and the second segment goes from $(10, 9)$ to $(10, 11)$.

The smallest possible slope is obtained by connecting $(0, 1)$ to $(10, 9)$, giving slope 0.8. The largest possible slope is obtained by connecting $(0, -1)$ to $(10, 11)$, giving slope 1.2.

Visualization of the first example:



Input 2

```
3
0 0 0.7
2 0 1.5
1 99.3 1
```

Output 2

```
-1
```

Explanation 2

No non-vertical line can intersect all three segments, because the middle segment is too high compared to the two outer segments.