Computers often break during an Informatics competition. Therefore, there are $k$ people in the competition hall who are responsible for the technical support. For the purpose of the task, we will call them *runners*.

The hall can be represented as the 2D space, where at the beginning of the competition we can choose where to place each *runner* as a pair of $(x, y)$. Every *runner* also has their own speed$ s$.

During the competition, exactly $n$ computers will break, each characterized by a pair $(x2, y2)$. 1 *runner* must be selected to go from their position $(x1, y1)$ to $(x2, y2)$ in time $s\*\sqrt{(x1-x2)^{2}+(y1-y2)^{2}}$ and fix the computer, which happens in a negligible amount of time.

Their new position will now be $(x2, y2)$.

Computers break in such a way that all *runners* are free at the time of computer break, i.e. the $i+1$-th computer will break only after the $i$-th computer is fixed.

We want to minimize the total amount of time a contestant sits with a broken computer, waiting for the respective *runner* to come and fix it.

**Input**

The first line of the **runners.in** file contains the numbers$ n$ and $k$. The next $k$ lines contain one number each – $s$ - the speed of the *runner*.

Each of the next $n$ lines contains two numbers $(x, y)$ for the position of the respective broken computer.

**Output**

On $k$ lines in the file **runners.out**, print 1 pair of numbers $(p, q)$ - the positions of the *runners* at the start of the competition. They must satisfy the constraints $1\leq p,q\leq 10^{9}$.

On the next $n$ lines, print 1 number each - the index of the *runner* that will repair the corresponding computer.

**Scoring**

For each test, let *minScore* be the smallest score among all participants' scores and *yourScore* be your score. You will be awarded $1-\sqrt{1-\frac{minScore+1}{yourScore+1}}$ multiplied by the amount of points for the test.

**Constraints**

$$n=100 000$$

$1.0\leq s\leq 10.0$*,*$ s$has at most 6 decimal digits.

$$1\leq x,y\leq 10^{9}$$

 **Time limit: 5 sec.**

 **Memory limit: 256 MB.**

The tests are distributed as follows:

|  |  |
| --- | --- |
| **Percentage** | $$k$$ |
| $$15\%$$ | $$1 \leq k\leq 10$$ |
| $$20\%$$ | $$10 \leq k\leq 100$$ |
| $$30\%$$ | $$100 \leq k\leq 1 000$$ |
| $$20\%$$ | $$1 000\leq k\leq 10 000$$ |
| $$15\%$$ | $$10 000\leq k\leq 100 000$$ |

**Sample test**

|  |  |
| --- | --- |
| **Input (runners.in)** | **Output (runners.out)** |
| 5 21.3000001.8000003 86 79 410 21 5 | 3 86 71 2 2 2 1 |

**Example explanation**

The sample test is only for an explanation, in all real tests $n=100 000$.

*Runner* 1 travels for 0 time to position $(3, 8)$.

*Runner* 2 travels for 0 time to position $(6, 7)$.

*Runner* 2 travels for $1.8\*\sqrt{18}$ time to position $\left(9, 4\right)$.

*Runner* 2 travels for $1.8\*\sqrt{5}$ time to position $(10, 2)$.

*Runner* 1 travels for $1.3\*\sqrt{13}$ time to position $(1, 5)$.

The total time in which a competitor is waiting with a broken computer is $1.8\*\sqrt{18}+1.8\*\sqrt{5}+1.3\*\sqrt{13}≈16.348892254$.

**Tests generation**

The numbers $k, s, x, y$ are randomly generated in the respective intervals that bound them (each number in the interval has an equal chance).