Problem 1 Educated Guesses (4%)

This is an interactive task.

In an unknown parallel universe, you are currently standing in front of a giant nuclear bomb. Should it explore, the world will be destroyed.

In order to defuse the bomb and save the world, you need to input the correct password (a non-negative integer between 1 and 2147483647).

The bomber has left you 35 chances of unsuccessful attempts. You noticed that, after an incorrect attempt, the machine will prompt you the relative order of your guess and the correct password.

- If your guess is smaller than the correct password, it prints the character "<" on the screen.
- If your guess is larger, it prints the character ">" instead.

Write a program to defuse the bomb and save the world!

Interaction

The following describes the interaction between your program and the jury program (the judge system).

- To make a guess on the password, print a line containing

  ```
  ?  %d
  ```

  That is, a question mark followed by a space and the integer to output.

  To ensure no I/O delay, flush the I/O buffer immediately with

  ```
  fflush(stdout);
  ```

- After printing your guess (to the `STDOUT` stream), read the response of the jury program from the `STDIN` stream. The response will be of the following forms:

  1. " `Correct.` "
     When you have made the correct guess, the judge will respond this message.

     Note that it contains one string token. In this case, your program should terminate immediately and you will get the "Correct" verdict.

Problem 1 continued on next page...

2. “ `Attempt Limits Exceeded!` ”

   When your program has made 35 incorrect attempts, the judge will respond you this message. (Note that it contains 3 string tokens in total.)

   In this case, your program should terminate immediately, and you will get the ”Wrong-Answer” verdict.

3. “ `<` ”

   When your program made an incorrect attempt and the guess is strictly smaller than the correct password, the judge will respond this message.
   (It contains 1 string token consisting of 1 character.)

   In this case, your program can proceed making further attempts.

4. “ `>` ”

   When your program made an incorrect attempt and the guess is no smaller than the correct answer, the judge will respond this message. (It contains 1 string token consisting of 1 character.)

   In this case, your program can proceed making further attempts.

Notes

Note that, the interaction is made via the standard I/O streams `STDIN` and `STDOUT`.

It is guaranteed that the judge does possess an integer password between 1 and 2147483647, and he will respond honestly according to the rule described above.

Problem 2 Best Left- Partner (4%)

This problem serves as a building block for BonusB.

Consider the following problem:

Given a sequence of n integer $b_1, b_2, \ldots, b_n$, where $b_i > 0$ and a target lower-bound $L > 0$, compute for each b_i its best left- partner, which is defined to be the largest integer ℓ_i, $1 \le \ell_i \le i$, such that

$$b_{\ell_i} + b_{\ell_i+1} + \ldots + b_i = \sum_{\ell_i \le j \le i} b_j \ge L.$$

If no such integer exists, ℓ_i is defined to be -1.

Input

The first line contains an integer n and L, $(2 \le n \le 10^5)$, the length of the sequence and the target lower-bound.

The second line contains n integers, the values of b_i, where $b_i > 0$ is strictly positive.

Output

Print n integers, the best left- partner for each b_i, separated by a space.

Example

Input

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

Output

```
-1 -1 1 2 3 6 7 7
```

Note

In the above example, since $\sum_{1 \le i \le 2} < L$, we have $\ell_1 = \ell_2 = -1$.

For the remaining i, ℓ_i is the largest integer such that $\sum_{\ell_i \le j \le i} b_i \ge L$.

Also note that, according to the definition,
one can easily show that ℓ_i will form a non-decreasing sequence.

Problem (BonusB) Maximum Density Segment

Consider the following problem:

Given a sequence of n integer pairs $(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)$, where $b_i > 0$, and a target lower-bound $L > 0$, compute the best segment (ℓ, r) of the sequence, where $1 \le \ell \le r \le n$, such that

1. $\sum_{\ell \le i \le r} b_i \ge L$, and
2. the density of the segment, defined as

$$\left(\sum_{\ell \le i \le r} a_i \right) \Big/ \left(\sum_{\ell \le i \le r} b_i \right) \quad = \quad \frac{a_\ell + a_{\ell+1} + \ldots + a_r}{b_\ell + b_{\ell+1} + \ldots + b_r} \quad ,$$

   is maximized among all possible segments satisfying Condition 1.

In other words, we want to compute a segment (ℓ, r) satisfying Condition 1 such that, for any segment (ℓ', r') that also satisfies Condition 1, we always have

$$\left(\sum_{\ell \le i \le r} a_i \right) \Big/ \left(\sum_{\ell \le i \le r} b_i \right) \quad \ge \quad \left(\sum_{\ell' \le i \le r'} a_i \right) \Big/ \left(\sum_{\ell' \le i \le r'} b_i \right) .$$

To solve this problem efficiently (yet approximately), You may want to use the property given at the end of this problem description. It may also be helpful to use the solutions of W7-1 and W7-2 as building blocks for this problem.

Input

The first line contains two integers n and L, ($2 \le n \le 10^5$), the length of the sequence and the target lower-bound.

Each of the next n lines contain two integers, a_i and b_i, where $b_i > 0$ is strictly positive.

Output

Print a double-precision floating-point number, the maximum density that can be achieved, in the first line. In the second line, print the end points, ℓ and r, of the segment that achieves this density.

If there are multiple answers, print any of them.

Note that, the relative error between the density you output and the optimal density should be no more than 10^{-6}, and the end points ℓ and r must satisfy $\ell < r$.

Problem 3 continued on next page. . .

Example 1

Input

```
8 5
3 1
7 2
5 1
4 2
5 1
17 6
16 4
4 2
```

Output

```
3.5
2 5
```

Note

You may want to use the following property.

Lemma 1. Let α^* denote the optimal density that can be achieved by any segment of length at least two. Let $\alpha \in \mathbb{R}$ be an arbitrary real number.

Define a sequence $\texttt{C} = \{ c_1, c_2, \ldots, c_n \}$, where $c_i := a_i - \alpha \cdot b_i$.

Let S be the segment with maximum sum (with respect to C) among all segments satisfying Condition 1. Then

$$\begin{cases} \texttt{S} > 0 \\ \texttt{S} = 0 \\ \texttt{S} < 0 \end{cases} \iff \begin{cases} \alpha < \alpha^* \\ \alpha = \alpha^* \\ \alpha > \alpha^* \end{cases}$$

In other words, by inspecting the value of S, you know the relative order of your guess α and the optimal value α^*.