BONGI

Stop what you are doing, a new very fun game has just been released: *Bongi*. The game is played using N balls. The ith ball has the positive integer a_i written on it.

You must divide the balls into K baskets, numbered 1 to K. Every ball must go into one of the baskets and every basket must contain at least one ball. The *value* of a basket is the sum of the integers on the balls in it. Your *score* is the geometric mean[1] of the values of the baskets.

Consider the following example with $K = 2$ baskets and $N = 5$ balls of value 12, 14, 16, 17 and 25. One possible solution is as follows:

- The first basket contains 16 and 17, and
- The second basket contains 25, 12 and 14.

The score of this solution is $(16 + 17)^{0.5} \cdot (25 + 12 + 14)^{0.5} \approx 41.02$.

You do not have to produce the maximum score possible, instead you are awarded points based on the score you are able to achieve.

This is an **output only** problem. You do not submit source code for this task. Instead, you are given a series of input files and must submit the corresponding output files.

Constraints

For all test cases:

- $1 \leq a_i \leq 10000$ for all i.
- $2 \leq K \leq N \leq 100$.

Input

- The first line of input contains the two integers N and K.
- The second line contains N integers $a_1, a_2, \ldots, a_N$.

Output

Output K lines, each containing a collection of integers, such that the ith line contains the values of the balls in the ith basket.

[1] The geometric mean of k values $a_1, a_2, ..., a_k$ is defined as $a_1^{1/k} \times a_2^{1/k} \times ... \times a_k^{1/k}$. For example, the geometric mean of 8, 9 and 10 is $8^{1/3} \times 9^{1/3} \times 10^{1/3} \approx 2 \times 2.080 \times 2.154 \approx 8.963$.

Scoring

If you do not put every ball into exactly one basket, and have each basket contain at least one ball, you will score 0% of the points for that test case.

Otherwise, let X be the score of your solution, B be the judges' best solution and $S = a_1 + a_2 + \cdots + a_N$. Then you will score

$$\min\left(100 \cdot \frac{10 + S/K - B}{10 + S/K - X}, 100\right)$$

percent of the points for that test case.

Note that it is possible to prove that $X \leq S/K$, thus the scoring function is well-defined.

Testcase	Points	N	K	Judges' best score
1	5	5	2	42
2	10	20	10	9564.453334019241
3	10	100	10	55237.69999809915
4	15	100	20	25324.1999731493
5	20	100	30	16980.161334096203
6	20	100	40	13886.860798648839
7	10	100	50	10086.367502580433
8	10	100	60	8392.505093895417

Submitting

To submit your output file for test case X, you must upload a file named `output_X.txt`. For example, for test case 4, the output file must be named `output_4.txt`.

You can choose to upload your output file for each test case individually, or together in a zip file.

Your score for this problem is the sum of your maximum scores for each test case, over all submissions you have made to this problem.

Sample Input 1

```
5 2
16 25 17 14 12
```

Sample Output 1

```
16 17
25 12 14
```

Explanation

The score of the solution is $X = (16+17)^{0.5} \cdot (25+12+14)^{0.5} \approx 41.02$. For this example, the best solution of the jury has score 42. Therefore, the percentage of points obtained by this output is

$$\approx 100 \cdot \frac{10 + 84/2 - 42}{10 + 84/2 - 41.02} \approx 91.11.$$

This example corresponds to the first test, which is worth 5 points. The number of points obtained when submitting this solution is 4.56/5.

Sample Code

You are provided with two files, `template.cpp` and `bongi.hpp`, to help you implement your solution.

The file `bongi.hpp` contains two functions that you may find useful:

```
double get_score(vector<vector<int> > solution);
```

This function returns the score of a solution. For example, if `solution = [[ 16, 17 ], [ 25, 12, 14 ]]`, then `get_score(solution)` returns (approximately) `41.02`.

```
double get_percentage(vector<vector<int> > solution);
```

This function returns the percentage of points that a solution would be given. For example, if `solution = [[ 16, 17 ], [ 25, 12, 14 ]]`, then `get_percentage(solution)` returns (approximately) `91.11`.

The file `template.cpp` contains an example usage of `get_percentage`. You can compile `template.cpp` using this command:

```
g++ -Wall -std=c++17 -O2 -o template template.cpp
```