NOI '16 P5 - Drinking Water

Time limit: 2.0s **Memory limit:** 256M

There is an array of n real numbers. The i-th real number is h_i. It is guaranteed that initially h_is are distinct.

In each operation, you may choose an arbitrary number of h_is and set the value of each chosen h_i to be the average of the chosen numbers. The chosen numbers do **not** need to be consecutive (i.e. you can choose h_1 and h_3 but not h_2). You may apply the operation for at most k times.

Compute the maximum possible h_1 in this case.

Input Specification

The first line of the input consists of three positive integers n, k, p denoting the length of the array, the maximum number of operations, and the required precision.

The next line consists of n positive integers. The i-th integer denotes h_i. It is guaranteed that h_is are distinct.
$1 \leq h_i \leq 10^5$.

Output Specification

Output a line with a real number denoting the maximum h_1 using at most k operations.

The answer can only contain a non-negative integer part, the decimal point, and the decimal part. The non-negative integer part is required, and it is unnecessary to add signs before the output. If there is a decimal part, the integral part and the decimal part are separated by a decimal point. If there is no decimal part, there shall be no decimal points.

In your output, there can be **at most** $2p$ digits after the decimal point. It is recommended that you should keep at least p digits in the decimal part. It is guaranteed that the absolute difference between the reference answer and the true answer is at most 10^{-2p}.

Your output is considered to be correct if and only if the **absolute** difference between your output and the reference answer is less than 10^{-p}.

If the absolute difference between your output and the reference answer is at least 10^{-p} but less than 10^{-5}, you may get 40% of the points of the test case.

Sample Input 1

```
3 1 3
1 4 3
```

Sample Output 1

```
2.666667
```

Explanation for Sample 1

There are five possibilities since we can use at most one operation (not counting the trivial operation when you just choose one number).

- Performing no operations: $h_1 = 1$.
- Choosing h_1, h_2: now $h_1 = h_2 = (1 + 4)/2 = 5/2$.
- Choosing h_1, h_3: now $h_1 = h_3 = (1 + 3)/2 = 2$.
- Choosing h_2, h_3: now $h_1 = 1$.
- Choosing h_1, h_2, h_3: now $h_1 = h_2 = h_3 = (1 + 4 + 3)/3 = 8/3$.

Sample Input 2

```
3 2 3
1 4 3
```

Sample Output 2

```
3.000000
```

Explanation for Sample 2

The optimal solution is to apply two operations: first, choose h_1 and h_3. Then choose h_1 and h_2.

Attachment Package

The samples are available here.

Sample 3

See `ex_drink3.in` and `ex_drink3.ans`.

Constraints

Test case	n	k	p

1	≤ 2	≤ 5	$= 5$
2	≤ 4		
3	≤ 4		
4	≤ 10	$= 1$	
5		$= 10^9$	
6		≤ 10	
7			
8	≤ 100	$= 1$	
9		$= 10^9$	$= 40$
10		$\leq 10^9$	
11			
12			
13	≤ 250		$= 100$
14	≤ 500		$= 200$
15	≤ 700		$= 300$
16			
17			
18	$\leq 2\,500$		$= 1\,000$
19	$\leq 4\,000$		$= 1\,500$
20	$\leq 8\,000$		$= 3\,000$

For all test cases, it is guaranteed that $3 \leq p \leq 3\,000$, $1 \leq n \leq 8\,000$, and $1 \leq k \leq 10^9$.

Hints

To guarantee precision, we need to keep more than p digits after the decimal point if possible when performing arithmetic operations. It can be shown that the reference solution to each subtask can guarantee the absolute difference between the output of the solution to the subtask and the reference answer is less than 10^{-p} when each arithmetic operation keeps at most $\frac{6}{5}p$ digits after the decimal point.

A library for high-precision floating number operations is provided here.

Using the library is optional.

The `Decimal` library supports high-precision floating point addition, subtraction, multiplication, division, comparisons, and conversions to and from `double`, integers, and strings. **Note**: Here, one operand for multiplication and division must be an integer. You cannot multiply a `double` or a `Decimal` with another `Decimal`.

The library defines a constant P, which says the absolute error of each operation is at most 10^{-P}. If you are using the C++ version, in the 9-th line of `drink_sample.cpp`, `const int PREC = 2100;` defines the constant P. If you are using the C version of the library, the line `#define PREC 2100` in the 9-th line of `drink_sample.c` defines the constant P. If you are using Pascal, `2100` in the 8-th and the 14-th lines of `drink_sample.pas` is the constant P. If you need to modify P, please change both occurrences to the same value.

The time complexity of any arithmetic operation provided by the library is bounded above by $\mathcal{O}(P)$.

The space complexity of any instance of the `Decimal` class is bounded above by $\mathcal{O}(P)$. More precisely, each instance should take at most $\frac{4P}{9} + 16$ bytes of space.

When calling a function provided by the `Decimal` class, the following conditions must be satisfied:

- In each intermediate step, a `Decimal` number has an absolute value of at most 10^{18}.
- `int/longint` parameters have absolute values of at most 10^9.
- `long long/int64` parameters have absolute values of at most 10^{18}.
- A `double` parameter must be a valid real number and has an absolute value of at most 10^9.
- An argument of type `string` must represent a valid real number. In other words, the string should begin with the sign part (a negative number has a minus sign, and a non-negative number has no signs), then a string consisting of digits representing the integer part, a decimal point, and a string consisting of digits representing the decimal part. The decimal part and the decimal point can be omitted simultaneously. The string should not represent a real number with absolute value greater than 10^9.
- You cannot divide by zero.
- When converting a `Decimal` to a string, the number of digits after the decimal point must be greater than zero. If you are using C, please make sure you've allocated enough space to store the string returned by `to_string_d(Decimal, char*, int)`.

The programs `drink_sample.cpp/c/pas` are empty programs except the library. You may work directly on the samples, but this is optional.

The programs `decimal_test.cpp/c/pas` are example programs for the `Decimal` library. You can use the program to learn more about the interfaces and the implementation of the library, as well as how to use each function of the library.