A. Non-zero

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Guy-Manuel and Thomas have an array a of n integers $[a_1, a_2, \dots, a_n]$. In one step they can add 1 to any element of the array. Formally, in one step they can choose any integer index i ($1 \le i \le n$) and do $a_i := a_i + 1$.

If either the sum or the product of all elements in the array is equal to zero, Guy-Manuel and Thomas do not mind to do this operation one more time.

What is the minimum number of steps they need to do to make both the sum and the product of all elements in the array **different from zero**? Formally, find the minimum number of steps to make $a_1 + a_2 + \dots + a_n \ne 0$ and $a_1 \cdot a_2 \cdot \dots \cdot a_n \ne 0$.

Input

Each test contains multiple test cases.

The first line contains the number of test cases t ($1 \le t \le 10^3$). The description of the test cases follows.

The first line of each test case contains an integer n ($1 \le n \le 100$) — the size of the array.

The second line of each test case contains n integers $a_1, a_2, \dots, a_n$ ($-100 \le a_i \le 100$) — elements of the array .

Output

For each test case, output the minimum number of steps required to make both sum and product of all elements in the array different from zero.

Example

input

```
4
3
2 -1 -1
4
-1 0 0 1
2
-1 2
3
0 -2 1
```

output

```
1
2
0
2
```

Note

In the first test case, the sum is 0. If we add 1 to the first element, the array will be $[3, -1, -1]$, the sum will be equal to 1 and the product will be equal to 3.

In the second test case, both product and sum are 0. If we add 1 to the second and the third element, the array will be $[-1, 1, 1, 1]$, the sum will be equal to 2 and the product will be equal to -1. It can be shown that fewer steps can't be enough.

In the third test case, both sum and product are non-zero, we don't need to do anything.

In the fourth test case, after adding 1 twice to the first element the array will be $[2, -2, 1]$, the sum will be 1 and the product will be -4.

B. Assigning to Classes

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Reminder: the median of the array $[a_1, a_2, \dots, a_{2k+1}]$ of odd number of elements is defined as follows: let $[b_1, b_2, \dots, b_{2k+1}]$ be the elements of the array in the sorted order. Then median of this array is equal to b_{k+1}.

There are $2n$ students, the i-th student has skill level a_i. It's **not guaranteed** that all skill levels are distinct.

Let's define **skill level of a class** as the median of skill levels of students of the class.

As a principal of the school, you would like to assign each student to one of the 2 classes such that each class has **odd number of students** (not divisible by 2). The number of students in the classes may be equal or different, by your choice. Every student has to be assigned to exactly one class. Among such partitions, you want to choose one in which the absolute difference between skill levels of the classes is minimized.

What is the minimum possible absolute difference you can achieve?

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 10^5$) — the number of students halved.

The second line of each test case contains $2n$ integers $a_1, a_2, \dots, a_{2n}$ ($1 \le a_i \le 10^9$) — skill levels of students.

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

For each test case, output a single integer, the minimum possible absolute difference between skill levels of two classes of odd sizes.

Example

input

```
3
1
1 1
3
6 5 4 1 2 3
5
13 4 20 13 2 5 8 3 17 16
```

output

```
0
1
5
```

Note

In the first test, there is only one way to partition students — one in each class. The absolute difference of the skill levels will be $|1 - 1| = 0$.

In the second test, one of the possible partitions is to make the first class of students with skill levels $[6, 4, 2]$, so that the skill level of the first class will be 4, and second with $[5, 1, 3]$, so that the skill level of the second class will be 3. Absolute difference will be $|4 - 3| = 1$.

Note that you can't assign like $[2, 3]$, $[6, 5, 4, 1]$ or $[]$, $[6, 5, 4, 1, 2, 3]$ because classes have even number of students.

$[2]$, $[1, 3, 4]$ is also not possible because students with skills 5 and 6 aren't assigned to a class.

In the third test you can assign the students in the following way: $[3, 4, 13, 13, 20], [2, 5, 8, 16, 17]$ or $[3, 8, 17], [2, 4, 5, 13, 13, 16, 20]$. Both divisions give minimal possible absolute difference.

C. Anu Has a Function

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Anu has created her own function f: $f(x, y) = (x | y) - y$ where $|$ denotes the bitwise OR operation. For example, $f(11, 6) = (11|6) - 6 = 15 - 6 = 9$. It can be proved that for any nonnegative numbers x and y value of $f(x, y)$ is also nonnegative.

She would like to research more about this function and has created multiple problems for herself. But she isn't able to solve all of them and needs your help. Here is one of these problems.

A value of an array $[a_1, a_2, \dots, a_n]$ is defined as $f(f(\dots f(f(a_1, a_2), a_3), \dots a_{n-1}), a_n)$ (see notes). You are given an array with **not necessarily distinct** elements. How should you reorder its elements so that the value of the array is maximal possible?

Input

The first line contains a single integer n ($1 \le n \le 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$). Elements of the array are **not guaranteed** to be different.

Output

Output n integers, the reordering of the array with maximum value. If there are multiple answers, print any.

Examples

input

```
4
4 0 11 6
```

output

```
11 6 4 0
```

input

```
1
13
```

output

```
13
```

Note

In the first testcase, value of the array $[11, 6, 4, 0]$ is $f(f(f(11, 6), 4), 0) = f(f(9, 4), 0) = f(9, 0) = 9$.

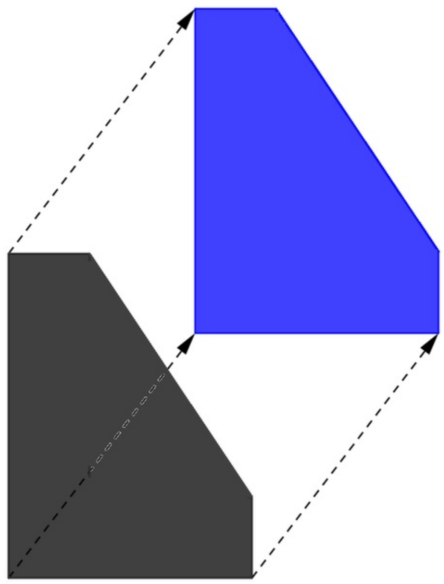
$[11, 4, 0, 6]$ is also a valid answer.

D. Aerodynamic

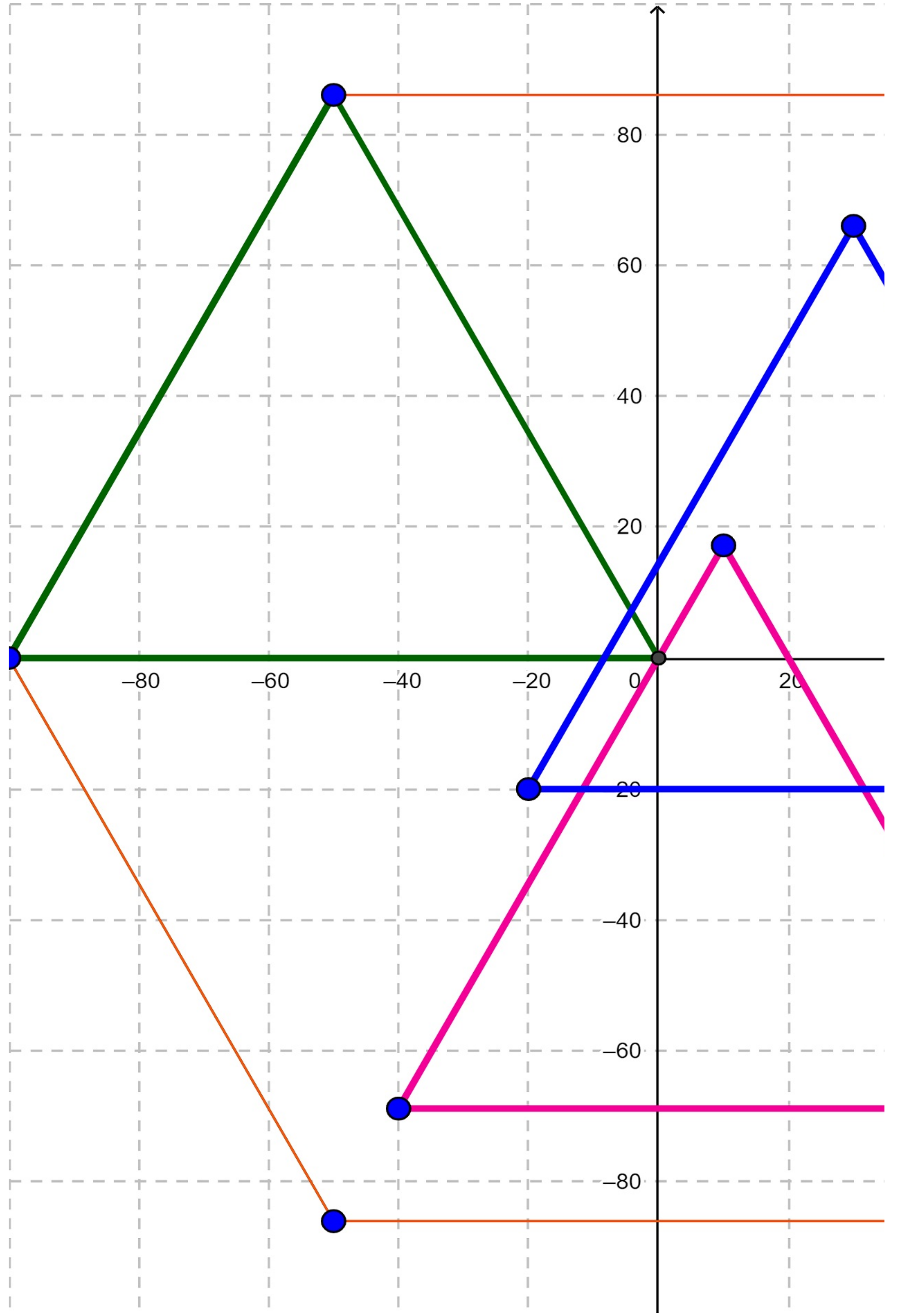
time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Guy-Manuel and Thomas are going to build a polygon spaceship.

You're given a **strictly convex** (i. e. no three points are collinear) polygon P which is defined by coordinates of its vertices. Define $P(x, y)$ as a polygon obtained by translating P by vector $\overrightarrow{(x, y)}$. The picture below depicts an example of the translation:



Define T as a set of points which is the union of all $P(x, y)$ such that the origin $(0, 0)$ lies in $P(x, y)$ (both strictly inside and on the boundary). There is also an equivalent definition: a point (x, y) lies in T only if there are two points A, B in P such that $\overrightarrow{AB} = \overrightarrow{(x, y)}$. One can prove T is a polygon too. For example, if P is a regular triangle then T is a regular hexagon. At the picture below P is drawn in black and some $P(x, y)$ which contain the origin are drawn in colored:



The spaceship has the best aerodynamic performance if P and T are similar. Your task is to check whether the polygons P and T are similar.

Input

The first line of input will contain a single integer n ($3 \le n \le 10^5$) — the number of points.

The i-th of the next n lines contains two integers x_i, y_i ($|x_i|, |y_i| \le 10^9$), denoting the coordinates of the i-th vertex.

It is guaranteed that these points are listed in counterclockwise order and these points form a strictly convex polygon.

Output

Output "YES" in a separate line, if P and T are similar. Otherwise, output "NO" in a separate line. You can print each letter in any case (upper or lower).

Examples

input

```
4
1 0
4 1
3 4
```

```
0 3
```

output

```
YES
```

input

```
3
100 86
50 0
150 0
```

output

```
nO
```

input

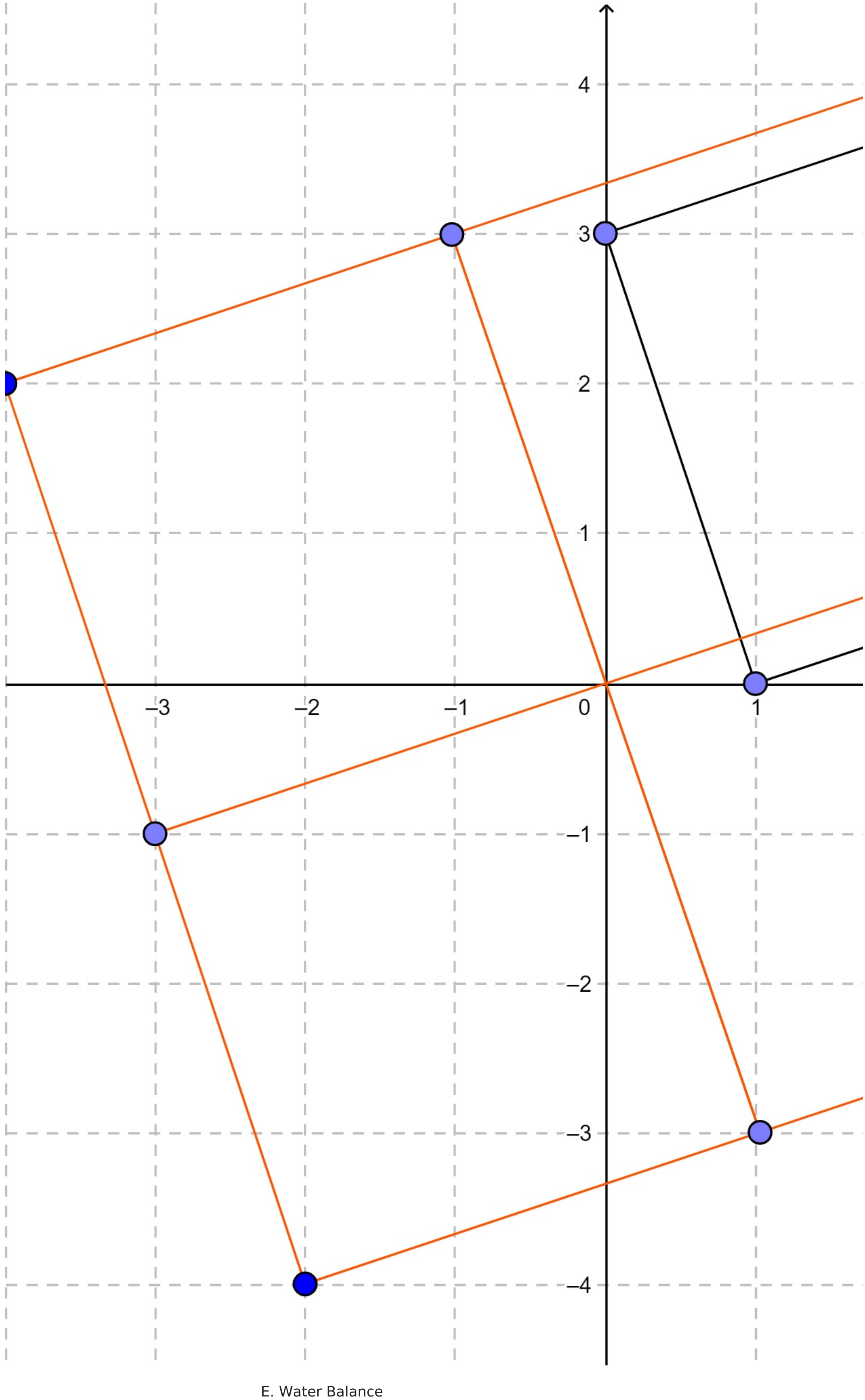
```
8
0 0
1 0
2 1
3 3
4 6
3 6
2 5
1 3
```

output

```
YES
```

Note

The following image shows the first sample: both P and T are squares. The second sample was shown in the statements.



E. Water Balance

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are n water tanks in a row, i-th of them contains a_i liters of water. The tanks are numbered from 1 to n from left to right.

You can perform the following operation: choose some subsegment $[l, r]$ ($1 \le l \le r \le n$), and redistribute water in tanks $l, l+1, \dots, r$ evenly. In other words, replace each of $a_l, a_{l+1}, \dots, a_r$ by $\frac{a_l + a_{l+1} + \dots + a_r}{r-l+1}$. For example, if for volumes $[1, 3, 6, 7]$ you choose $l = 2, r = 3$, new volumes of water will be $[1, 4.5, 4.5, 7]$. **You can perform this operation any number of times**.

What is the lexicographically smallest sequence of volumes of water that you can achieve?

As a reminder:

A sequence a is lexicographically smaller than a sequence b of the same length if and only if the following holds: in the first (leftmost) position where a and b differ, the sequence a has a smaller element than the corresponding element in b.

Input

The first line contains an integer n ($1 \le n \le 10^6$) — the number of water tanks.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^6$) — initial volumes of water in the water tanks, in liters.

Because of large input, reading input as doubles **is not recommended**.

Output

Print the lexicographically smallest sequence you can get. In the i-th line print the final volume of water in the i-th tank.

Your answer is considered correct if the absolute or relative error of each a_i does not exceed 10^{-9}.

Formally, let your answer be $a_1, a_2, \dots, a_n$, and the jury's answer be $b_1, b_2, \dots, b_n$. Your answer is accepted if and only if $\frac{|a_i - b_i|}{\max(1, |b_i|)} \le 10^{-9}$ for each i.

Examples

input

```
4
7 5 5 7
```

output

```
5.666666667
5.666666667
5.666666667
7.000000000
```

input

```
5
7 8 8 10 12
```

output

```
7.000000000
8.000000000
8.000000000
10.000000000
12.000000000
```

input

```
10
3 9 5 5 1 7 5 3 8 7
```

output

```
3.000000000
5.000000000
5.000000000
5.000000000
5.000000000
5.000000000
5.000000000
5.000000000
7.500000000
7.500000000
```

Note

In the first sample, you can get the sequence by applying the operation for subsegment $[1, 3]$.

In the second sample, you can't get any lexicographically smaller sequence.