ECI 2025 Wellcome contest

A. Is it rated - 2

time limit per test: 1 second
memory limit per test: 256 megabytes

Interaction

This is an interactive problem. You need to read participants' queries from standard input and print your responses to standard output. You don't know the number of queries upfront, so you'll need to process them as you get them; you'll know you're done once you reach the end of the file.

In each query, you will be asked the question, written in one line. You have to answer it correctly, patiently and without any display of emotions. Your response is case-insensitive.

Please make sure to use the stream flushing operation after each response in order not to leave part of your output in some buffer.

Example

input

```
Is it rated?
Is it rated?
Is it rated?
```

output

```
NO
NO
NO
```

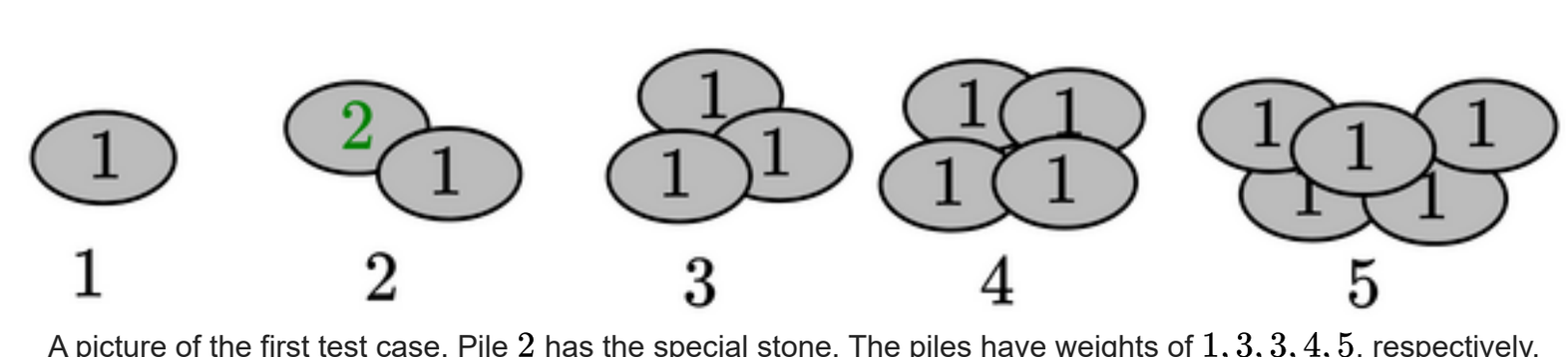
B. Interview

time limit per test: 2 seconds
memory limit per test: 256 megabytes

This is an interactive problem. If you are unsure how interactive problems work, then it is recommended to read the guide for participants.

Before the last stage of the exam, the director conducted an interview. He gave Gon n piles of stones, the i-th pile having a_i stones.

Each stone is identical and weighs 1 grams, except for one special stone that is part of an unknown pile and weighs 2 grams.



A picture of the first test case. Pile 2 has the special stone. The piles have weights of $1, 3, 3, 4, 5$, respectively.

Gon can only ask the director questions of one kind: he can choose k piles, and the director will tell him the total weight of the piles chosen. More formally, Gon can choose an integer k ($1 \le k \le n$) and k unique piles $p_1, p_2, \dots, p_k$ ($1 \le p_i \le n$), and the director will return the total weight $m_{p_1} + m_{p_2} + \dots + m_{p_k}$, where m_i denotes the weight of pile i.

Gon is tasked with finding the pile that contains the special stone. However, the director is busy. Help Gon find this pile in at most $\mathbf{30}$ queries.

Input

The input data contains several test cases. The first line contains one integer t ($1 \le t \le 1000$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the number of piles.

The second line of each test case contains n integers a_i ($1 \le a_i \le 10^4$) — the number of stones in each pile.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

After reading the input for each test case, proceed with the interaction as follows.

Interaction

You can perform the operation at most $\mathbf{30}$ times to guess the pile.

To make a guess, print a line with the following format:

- **?** $k\ p_1\ p_2\ p_3\ \dots\ p_{k-1}\ p_k$ ($1 \le k \le n$; $1 \le p_i \le n$; all p_i are distinct) — the indices of the piles.

After each operation, you should read a line containing a single integer x — the sum of weights of the chosen piles. (Formally, $x = m_{p_1} + m_{p_2} + \dots + m_{p_k}$.)
When you know the index of the pile with the special stone, print one line in the following format: **!** m ($1 \le m \le n$).

After that, move on to the next test case, or terminate the program if there are no more test cases remaining.

If your program performs more than 30 operations for one test case or makes an invalid query, you may receive a `Wrong Answer` verdict.

After you print a query or the answer, please remember to output the end of the line and flush the output. Otherwise, you may get `Idleness limit exceeded` or some other verdict. To do this, use the following:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- see the documentation for other languages.

It is additionally recommended to read the interactive problems guide for participants.

Hacks

To make a hack, use the following format.

The first line should contain a single integer t ($1 \le t \le 1000$) — the number of test cases.

The first line of each test case should contain two integers n, m ($1 \le n \le 2 \cdot 10^5$) – the number of piles and the pile with the special stone.

The second line of each test case should contain n integers a_i ($1 \le a_i \le 10^4$) — the number of stones in each pile.

Note that the interactor is **not** adaptive, meaning that the answer is known before the participant asks the queries and doesn't depend on the queries asked by the participant.

Example

input

```
2
5
1 2 3 4 5

11

6

3

7
1 2 3 5 3 4 2

12

6
```

output

```
? 4 1 2 3 4

? 2 2 3

? 1 2

! 2

? 4 2 3 5 6

? 2 1 4

! 7
```

Note

In the first test case, the stone with weight two is located in pile 2, as shown in the picture. We perform the following interaction:

- `? 4 1 2 3 4` — ask the total weight of piles 1, 2, 3, and 4. The total weight we receive back is $1+3+3+4=11$.
- `? 2 2 3` — ask the total weight of piles 2 and 3. The total weight we receive back is $3+3=6$.
- `? 1 2` — ask the total weight of pile 2. The total weight we receive back is 3.
- `! 2` — we have figured out that pile 2 contains the special stone, so we output it and move on to the next test case.

In the second test case, the stone with weight two is located on index 7. We perform the following interaction:

- `? 4 2 3 5 6` — ask the total weight of piles 2, 3, 5, and 6. The total weight we receive back is $2+3+3+4=12$.
- `? 2 1 4` — ask the total weight of piles 1 and 4. The total weight we receive back is $1+5=6$.
- `! 7` — we have somehow figured out that pile 7 contains the special stone, so we output it and end the interaction.

C. Bear and Prime 100

time limit per test: 1 second
memory limit per test: 256 megabytes

This is an interactive problem. In the output section below you will see the information about flushing the output.

Bear Limak thinks of some hidden number — an integer from interval $[2, 100]$. Your task is to say if the hidden number is prime or composite.

Integer $x > 1$ is called prime if it has exactly two distinct divisors, 1 and x. If integer $x > 1$ is not prime, it's called composite.

You can ask up to 20 queries about divisors of the hidden number. In each query you should print an integer from interval $[2, 100]$. The system will answer "`yes`" if your integer is a divisor of the hidden number. Otherwise, the answer will be "`no`".

For example, if the hidden number is 14 then the system will answer "`yes`" only if you print 2, 7 or 14.

When you are done asking queries, print "`prime`" or "`composite`" and terminate your program.

You will get the `Wrong Answer` verdict if you ask more than 20 queries, or if you print an integer not from the range $[2, 100]$. Also, you will get the `Wrong Answer` verdict if the printed answer isn't correct.

You will get the `Idleness Limit Exceeded` verdict if you don't print anything (but you should) or if you forget about flushing the output (more info below).

Input

After each query you should read one string from the input. It will be "`yes`" if the printed integer is a divisor of the hidden number, and "`no`" otherwise.

Output

Up to 20 times you can ask a query — print an integer from interval $[2, 100]$ in one line. You have to both print the end-of-line character and `flush` the output. After flushing you should read a response from the input.

In any moment you can print the answer "`prime`" or "`composite`" (without the quotes). After that, flush the output and terminate your program.

To flush you can use (just after printing an integer and end-of-line):

- `fflush(stdout)` in C++;
- `System.out.flush()` in Java;
- `stdout.flush()` in Python;
- `flush(output)` in Pascal;
- See the documentation for other languages.

Hacking. To hack someone, as the input you should print the hidden number — one integer from the interval $[2, 100]$. Of course, his/her solution won't be able to read the hidden number from the input.

Examples

input

```
yes
no
yes
```

output

```
2
80
5
composite
```

input

```
no
yes
no
no
no
```

output

```
58
59
78
78
2
prime
```

Note

The hidden number in the first query is 30. In a table below you can see a better form of the provided example of the communication process.

solution	system
2	yes
80	no
5	yes
composite	

The hidden number is divisible by both 2 and 5. Thus, it must be composite. Note that it isn't necessary to know the exact value of the hidden number. In this test, the hidden number is 30.

solution	system
58	no
59	yes
78	no
78	no
2	no
prime	

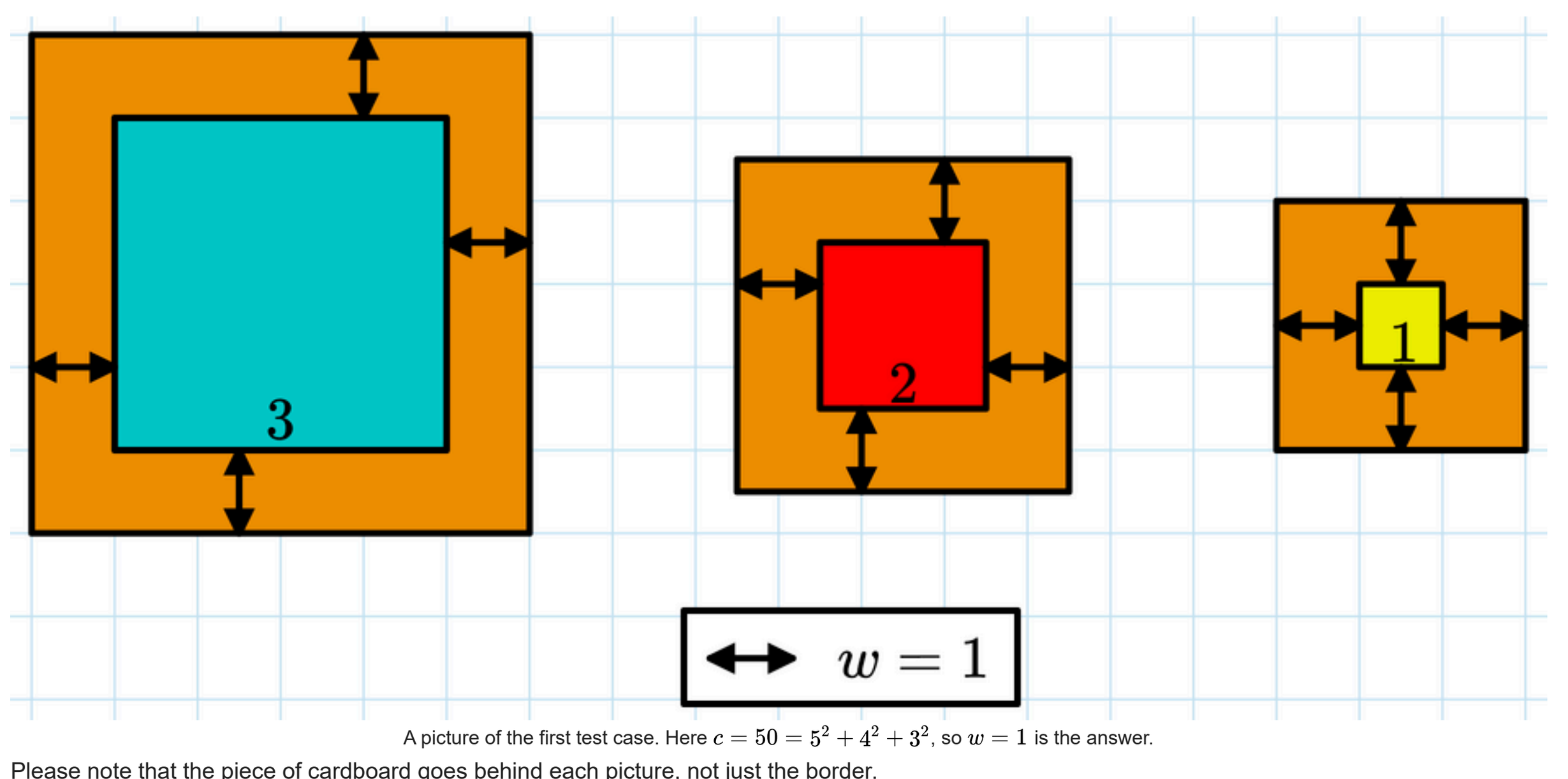
59 is a divisor of the hidden number. In the interval $[2, 100]$ there is only one number with this divisor. The hidden number must be 59, which is prime. Note that the answer is known even after the second query and you could print it then and terminate. Though, it isn't forbidden to ask unnecessary queries (unless you exceed the limit of 20 queries).

D. Cardboard for Pictures

time limit per test: 2 seconds
memory limit per test: 256 megabytes

Mircea has n pictures. The i-th picture is a square with a side length of s_i centimeters.

He mounted each picture on a square piece of cardboard so that each picture has a border of w centimeters of cardboard on all sides. In total, he used c square centimeters of cardboard. Given the picture sizes and the value c, can you find the value of w?



A picture of the first test case. Here $c = 50 = 5^2 + 4^2 + 3^2$, so $w = 1$ is the answer.

Please note that the piece of cardboard goes behind each picture, not just the border.

Input

The first line contains a single integer t ($1 \le t \le 1000$) — the number of test cases.

The first line of each test case contains two positive integers n ($1 \le n \le 2 \cdot 10^5$) and c ($1 \le c \le 10^{18}$) — the number of paintings, and the amount of used square centimeters of cardboard.

The second line of each test case contains n space-separated integers s_i ($1 \le s_i \le 10^4$) — the sizes of the paintings.

The sum of n over all test cases doesn't exceed $2 \cdot 10^5$.

Additional constraint on the input: Such an integer w exists for each test case.

Please note, that some of the input for some test cases won't fit into 32-bit integer type, so you should use at least 64-bit integer type in your programming language (like `long long` for C++).

Output

For each test case, output a single integer — the value of w ($w \ge 1$) which was used to use exactly c squared centimeters of cardboard.

Example

input

```
10
3 50
3 2 1
1 100
6
5 500
2 2 2 2 2
2 365
3 4
2 469077255466389
10000 2023
10 635472106413848880
9181 4243 7777 1859 2017 4397 14 9390 2245 7225
7 176345687772781240
9202 9407 9229 6257 7743 5738 7966
14 865563946464579627
3654 5483 1657 7571 1639 9815 122 9468 3079 2666 5498 4540 7861 5384
19 977162053008871403
9169 9520 9209 9013 9300 9843 9933 9454 9960 9167 9964 9701 9251 9404 9462 9277 9661 9164 9161
18 886531871815571953
2609 10 5098 9591 949 8485 6385 4586 1064 5412 6564 8460 2245 6552 5089 8353 3803 3764
```

output

```
1
2
4
5
7654321
126040443
79356352
124321725
113385729
110961227
```

Note

The first test case is explained in the statement.

For the second test case, the chosen w was 2, thus the only cardboard covers an area of $c = (2 \cdot 2 + 6)^2 = 10^2 = 100$ squared centimeters.

For the third test case, the chosen w was 4, which obtains the covered area $c = (2 \cdot 4 + 2)^2 \times 5 = 10^2 \times 5 = 100 \times 5 = 500$ squared centimeters.

E. Books

time limit per test: 2 seconds
memory limit per test: 256 megabytes

When Valera has got some free time, he goes to the library to read some books. Today he's got t free minutes to read. That's why Valera took n books in the library and for each book he estimated the time he is going to need to read it. Let's number the books by integers from 1 to n. Valera needs a_i minutes to read the i-th book.

Valera decided to choose an arbitrary book with number i and read the books one by one, starting from this book. In other words, he will first read book number i, then book number $i+1$, then book number $i+2$ and so on. He continues the process until he either runs out of the free time or finishes reading the n-th book. Valera reads each book up to the end, that is, he doesn't start reading the book if he doesn't have enough free time to finish reading it.

Print the maximum number of books Valera can read.

Input

The first line contains two integers n and t ($1 \le n \le 10^5$; $1 \le t \le 10^9$) — the number of books and the number of free minutes Valera's got. The second line contains a sequence of n integers $a_1, a_2, ..., a_n$ ($1 \le a_i \le 10^4$), where number a_i shows the number of minutes that the boy needs to read the i-th book.

Output

Print a single integer — the maximum number of books Valera can read.

Examples

input

```
4 5
3 1 2 1
```

output

```
3
```

input

```
3 3
2 2 3
```

output

```
1
```

F. Money Trees

time limit per test: 2 seconds
memory limit per test: 256 megabytes

Luca is in front of a row of n trees. The i-th tree has a_i fruit and height h_i.

He wants to choose a contiguous subarray of the array $[h_l, h_{l+1}, \dots, h_r]$ such that for each i ($l \le i < r$), h_i **is divisible**† **by** h_{i+1}. He will collect all the fruit from each of the trees in the subarray (that is, he will collect $a_l + a_{l+1} + \dots + a_r$ fruits). However, if he collects more than k fruits in total, he will get caught.

What is the maximum length of a subarray Luca can choose so he doesn't get caught?

† x is *divisible* by y if the ratio $\frac{x}{y}$ is an integer.

Input

The first line contains a single integer t ($1 \le t \le 1000$) — the number of test cases.

The first of each test case line contains two space-separated integers n and k ($1 \le n \le 2 \cdot 10^5$; $1 \le k \le 10^9$) — the number of trees and the maximum amount of fruits Luca can collect without getting caught.

The second line of each test case contains n space-separated integers a_i ($1 \le a_i \le 10^4$) — the number of fruits in the i-th tree.

The third line of each test case contains n space-separated integers h_i ($1 \le h_i \le 10^9$) — the height of the i-th tree.

The sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case output a single integer, the length of the maximum length contiguous subarray satisfying the conditions, or 0 if there is no such subarray.

Example

input

```
5
5 12
3 2 4 1 8
4 4 2 4 1
4 8
5 4 1 2
6 2 3 1
3 12
7 9 10
2 2 4
1 10
11
1
7 10
2 6 3 1 5 10 6
72 24 24 12 4 4 2
```

output

```
3
2
1
0
3
```

Note

In the first test case, Luca can select the subarray with $l = 1$ and $r = 3$.

In the second test case, Luca can select the subarray with $l = 3$ and $r = 4$.

In the third test case, Luca can select the subarray with $l = 2$ and $r = 2$.

G. Pair of Topics

time limit per test: 2 seconds
memory limit per test: 256 megabytes

The next lecture in a high school requires two topics to be discussed. The i-th topic is interesting by a_i units for the teacher and by b_i units for the students.

The pair of topics i and j ($i < j$) is called **good** if $a_i + a_j > b_i + b_j$ (i.e. it is more interesting for the teacher).

Your task is to find the number of **good** pairs of topics.

Input

The first line of the input contains one integer n ($2 \le n \le 2 \cdot 10^5$) — the number of topics.

The second line of the input contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^9$), where a_i is the interestingness of the i-th topic for the teacher.

The third line of the input contains n integers $b_1, b_2, \dots, b_n$ ($1 \le b_i \le 10^9$), where b_i is the interestingness of the i-th topic for the students.

Output

Print one integer — the number of **good** pairs of topic.

Examples

input

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

output

```
7
```

input

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

output

```
0
```