B. Applejack and Storages

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

This year in Equestria was a year of plenty, so Applejack has decided to build some new apple storages. According to the advice of the farm designers, she chose to build two storages with non-zero area: one in the shape of a square and another one in the shape of a rectangle (which possibly can be a square as well).

Applejack will build the storages using planks, she is going to spend exactly one plank on each side of the storage. She can get planks from her friend's company. Initially, the company storehouse has n planks, Applejack knows their lengths. The company keeps working so it receives orders and orders the planks itself. Applejack's friend can provide her with information about each operation. For convenience, he will give her information according to the following format:

- $+ x$: the storehouse received a plank with length x
- $- x$: one plank with length x was removed from the storehouse (it is guaranteed that the storehouse had some planks with length x).

Applejack is still unsure about when she is going to order the planks so she wants to know if she can order the planks to build rectangular and square storages out of them after every event at the storehouse. Applejack is busy collecting apples and she has completely no time to do the calculations so she asked you for help!

We remind you that all four sides of a square are equal, and a rectangle has two pairs of equal sides.

Input

The first line contains a single integer n ($1 \le n \le 10^5$): the initial amount of planks at the company's storehouse, the second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^5$): the lengths of the planks.

The third line contains a single integer q ($1 \le q \le 10^5$): the number of events in the company. Each of the next q lines contains a description of the events in a given format: the type of the event (a symbol $+$ or $-$) is given first, then goes the integer x ($1 \le x \le 10^5$).

Output

After every event in the company, print "YES" if two storages of the required shape can be built from the planks of that company's set, and print "NO" otherwise. You can print each letter in any case (upper or lower).

Example

input

Copy

```
6
1 1 1 2 1 1
6
+ 2
+ 1
```

```
- 1
+ 2
- 1
+ 2
```

output

Copy

```
NO
YES
NO
NO
NO
YES
```

Note

After the second event Applejack can build a rectangular storage using planks with lengths $1,2,1,2$ and a square storage using planks with lengths $1, 1, 1, 1$.

After the sixth event Applejack can build a rectangular storage using planks with lengths $2,2,2,2$ and a square storage using planks with lengths $1, 1, 1, 1$.

B. Arrays Sum

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

You are given a **non-decreasing** array of **non-negative** integers $a_1,a_2,\ldots,a_n$. Also you are given a positive integer k.

You want to find m **non-decreasing** arrays of **non-negative** integers $b_1,b_2,\ldots,b_m$, such that:

- The size of b_i is equal to n for all $1\le i\le m$.
- For all $1\le j\le n$, $a_j=b_{1,j}+b_{2,j}+\ldots+b_{m,j}$. In the other word, array a is the sum of arrays b_i.
- The number of different elements in the array b_i is at most k for all $1\le i\le m$.

Find the minimum possible value of m, or report that there is no possible m.

Input

The first line contains one integer t ($1\le t\le 100$): the number of test cases.

The first line of each test case contains two integers n, k ($1\le n\le 100$, $1\le k\le n$).

The second line contains n integers $a_1,a_2,\ldots,a_n$ ($0\le a_1\le a_2\le\ldots\le a_n\le 100$, $a_n>0$).

Output

For each test case print a single integer: the minimum possible value of m. If there is no such m, print -1.

Example

input

Copy

```
6
4 1
0 0 0 1
3 1
3 3 3
11 3
0 1 2 2 3 3 3 4 4 4 4
5 3
1 2 3 4 5
9 4
2 2 3 5 7 11 13 13 17
10 7
0 1 1 2 3 3 4 5 5 6
```

output

Copy

```
-1
1
2
2
2
1
```

Note

In the first test case, there is no possible m, because all elements of all arrays should be equal to 0. But in this case, it is impossible to get $a_4=1$ as the sum of zeros.

In the second test case, we can take $b_1=[3,3,3]$. 1 is the smallest possible value of m.

In the third test case, we can take $b_1=[0,1,1,1,2,2,2,2,2,2,2]$ and $b_2=[0,0,1,1,1,1,1,2,2,2,2]$. It's easy to see, that $a_i=b_{1,i}+b_{2,i}$ for all i and the number of different elements in b_1 and in b_2 is equal to 3 (so it is at most 3). It can be proven that 2 is the smallest possible value of m.

B. Universal Solution

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

Recently, you found a bot to play "Rock paper scissors" with. Unfortunately, the bot uses quite a simple algorithm to play: he has a string $s=s_1s_2...s_n$ of length n where each letter is either R, S or P.

While initializing, the bot is choosing a starting index pos ($1 \le pos \le n$), and then it can play any number of rounds. In the first round, he chooses "Rock", "Scissors" or "Paper" based on the value of s_{pos}:

- if s_{pos} is equal to R the bot chooses "Rock";
- if s_{pos} is equal to S the bot chooses "Scissors";
- if s_{pos} is equal to P the bot chooses "Paper";

In the second round, the bot's choice is based on the value of s_{pos+1}. In the third round — on s_{pos+2} and so on. After s_n the bot returns to s_1 and continues his game.

You plan to play n rounds and you've already figured out the string s but still don't know what is the starting index pos. But since the bot's tactic is so boring, you've decided to find n choices to each round to maximize the average number of wins.

In other words, let's suggest your choices are $c_1c_2...c_n$ and if the bot starts from index pos then you'll win in $win(pos)$ rounds. Find $c_1c_2...c_n$ such that $win(1)+win(2)+\cdots+win(n)$ is maximum possible.

Input

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

Next t lines contain test cases — one per line. The first and only line of each test case contains string $s=s_1s_2...s_n$ ($1 \le n \le 2 \cdot 10^5$; $s_i \in \{R,S,P\}$) — the string of the bot.

It's guaranteed that the total length of all strings in one test doesn't exceed $2 \cdot 10^5$.

Output

For each test case, print n choices $c_1c_2...c_n$ to maximize the average number of wins. Print them in the same manner as the string s.

If there are multiple optimal answers, print any of them.

Example

input

Copy

```
3
RRRR
RSP
S
```

output

Copy

```
PPPP
RSP
R
```

Note

In the first test case, the bot (wherever it starts) will always choose "Rock", so we can always choose "Paper". So, in any case, we will win all $n=4$ rounds, so the average is also equal to 4.

In the second test case:

- if bot will start from $pos=1$, then (s_1,c_1) is draw, (s_2,c_2) is draw and (s_3,c_3) is draw, so $win(1)=0$;

- if bot will start from $pos=2$, then (s_2,c_1) is win, (s_3,c_2) is win and (s_1,c_3) is win, so $win(2)=3$;
- if bot will start from $pos=3$, then (s_3,c_1) is lose, (s_1,c_2) is lose and (s_2,c_3) is lose, so $win(3)=0$;

The average is equal to $0+3+03=1$ and it can be proven that it's the maximum possible average.
A picture from Wikipedia explaining "Rock paper scissors" game:

B. Orac and Models

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

There are n models in the shop numbered from 1 to n, with sizes $s_1,s_2,\ldots,s_n$.
Orac will buy some of the models and will arrange them in the order of increasing numbers (i.e. indices, but not sizes).
Orac thinks that the obtained arrangement is **beatiful**, if for any two adjacent models with indices i_j and i_{j+1} (note that $i_j<i_{j+1}$, because Orac arranged them properly), i_{j+1} is divisible by i_j and $s_{i_j}<s_{i_{j+1}}$.
For example, for 6 models with sizes $\{3,6,7,7,7,7\}$, he can buy models with indices 1, 2, and 6, and the obtained arrangement will be beautiful. Also, note that the arrangement with exactly one model is also considered beautiful.
Orac wants to know the maximum number of models that he can buy, and he may ask you these queries many times.

Input

The first line contains one integer t $(1\le t\le 100)$: the number of queries.
Each query contains two lines. The first line contains one integer n $(1\le n\le 100000)$: the number of models in the shop, and the second line contains n integers $s_1,\ldots,s_n$ $(1\le s_i\le 10^9)$: the sizes of models.
It is guaranteed that the total sum of n is at most 100000.

Output

Print t lines, the i-th of them should contain the maximum number of models that Orac can buy for the i-th query.

Example

input

Copy

```
4
4
```

```
5 3 4 6
7
1 4 2 3 6 4 9
5
5 4 3 2 1
1
9
```

output

Copy

```
2
3
1
1
```

Note

In the first query, for example, Orac can buy models with indices 2 and 4, the arrangement will be beautiful because 4 is divisible by 2 and 6 is more than 3. By enumerating, we can easily find that there are no beautiful arrangements with more than two models.

In the second query, Orac can buy models with indices 1, 3, and 6. By enumerating, we can easily find that there are no beautiful arrangements with more than three models.

In the third query, there are no beautiful arrangements with more than one model.

B. Find The Array

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

You are given an array $[a_1,a_2,\ldots,a_n]$ such that $1 \le a_i \le 10^9$. Let S be the sum of all elements of the array a.

Let's call an array b of n integers **beautiful** if:

- $1 \le b_i \le 10^9$ for each i from 1 to n;
- for every pair of adjacent integers from the array (b_i,b_{i+1}), either b_i divides b_{i+1}, or b_{i+1} divides b_i (or both);
- $2\sum_{i=1}^{n}|a_i-b_i| \le S$.

Your task is to find any beautiful array. It can be shown that at least one beautiful array always exists.

Input

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

Each test case consists of two lines. The first line contains one integer n ($2 \le n \le 50$).

The second line contains n integers $a_1,a_2,\ldots,a_n$ ($1 \le a_i \le 10^9$).

Output

For each test case, print the beautiful array $b_1,b_2,\ldots,b_n$ ($1 \le b_i \le 10^9$) on a separate line. It can be shown that at least one beautiful array exists under these circumstances. If there are multiple answers, print any of them.

Example

input

Copy

```
4
5
1 2 3 4 5
2
4 6
2
1 1000000000
6
3 4 8 1 2 3
```

output

Copy

```
3 3 3 3 3
3 6
1 1000000000
4 4 8 1 3 3
```

B. Move and Turn

time limit per test
2 seconds
memory limit per test
512 megabytes
input
standard input
output
standard output

A robot is standing at the origin of the infinite two-dimensional plane. Each second the robot moves exactly 1 meter in one of the four cardinal directions: north, south, west, and east. For the first step the robot **can choose any of the four directions**, but then at the end of every second it **has to turn** 90 degrees left or right with respect to the direction it just moved in. For example, if the robot has just moved north or south, the next step it takes has to be either west or east, and vice versa.

The robot makes **exactly** n steps from its starting position according to the rules above. How many different points can the robot arrive to at the end? The final orientation of the robot can be ignored.

Input

The only line contains a single integer n ($1 \le n \le 1000$) — the number of steps the robot makes.

Output

Print a single integer — the number of different possible locations after **exactly** n steps.

Examples

input

Copy

```
1
```

output

Copy

```
4
```

input

Copy

```
2
```

output

Copy

```
4
```

input

Copy

```
3
```

output

Copy

```
12
```

Note

In the first sample case, the robot will end up 1 meter north, south, west, or east depending on its initial direction.

In the second sample case, the robot will always end up $2-\sqrt{}$ meters north-west, north-east, south-west, or south-east.

B. Bus of Characters

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

In the Bus of Characters there are n rows of seat, each having 2 seats. The width of both seats in the i-th row is w_i centimeters. All integers w_i are distinct.

Initially the bus is empty. On each of $2n$ stops one passenger enters the bus. There are two types of passengers:

- an introvert always chooses a row where both seats are empty. Among these rows he chooses the one with the smallest seats width and takes one of the seats in it;

- an extrovert always chooses a row where exactly one seat is occupied (by an introvert). Among these rows he chooses the one with the largest seats width and takes the vacant place in it.

  You are given the seats width in each row and the order the passengers enter the bus. Determine which row each passenger will take.

  ## Input

  The first line contains a single integer n ($1 \le n \le 200000$) — the number of rows in the bus.

  The second line contains the sequence of integers $w_1, w_2, \ldots, w_n$ ($1 \le w_i \le 10^9$), where w_i is the width of each of the seats in the i-th row. It is guaranteed that all w_i are **distinct**.

  The third line contains a string of length $2n$, consisting of digits '0' and '1' — the description of the order the passengers enter the bus. If the j-th character is '0', then the passenger that enters the bus on the j-th stop is an introvert. If the j-th character is '1', the the passenger that enters the bus on the j-th stop is an extrovert. It is guaranteed that the number of extroverts **equals** the number of introverts (i. e. both numbers equal n), and for each extrovert there **always** is a suitable row.

  ## Output

  Print $2n$ integers — the rows the passengers will take. The order of passengers should be the same as in input.

  Examples

  input

  Copy

  ```
  2
  3 1
  0011
  ```

  output

  Copy

  ```
  2 1 1 2
  ```

  input

  Copy

  ```
  6
  10 8 9 11 13 5
  010010011101
  ```

  output

  Copy

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

  ## Note

  In the first example the first passenger (introvert) chooses the row 2, because it has the seats with smallest width. The second passenger (introvert) chooses the row 1, because it is the only empty row now. The third passenger (extrovert) chooses the row 1, because it has exactly one occupied seat and the seat width is the largest among such rows. The fourth passenger (extrovert) chooses the row 2, because it is the only row with an empty place.

B. Knights of a Polygonal Table

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

Unlike Knights of a Round Table, Knights of a Polygonal Table deprived of nobility and happy to kill each other. But each knight has some power and a knight can kill another knight if and only if his power is greater than the power of victim. However, even such a knight will torment his conscience, so he can kill no more than k other knights. Also, each knight has some number of coins. After a kill, a knight can pick up all victim's coins.
Now each knight ponders: how many coins he can have if only he kills other knights?
You should answer this question for each knight.

Input

The first line contains two integers n and k ($1 \le n \le 10^5, 0 \le k \le \min(n-1, 10)$) — the number of knights and the number k from the statement.

The second line contains n integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le 10^9$) — powers of the knights. All p_i are distinct.

The third line contains n integers $c_1, c_2, \ldots, c_n$ ($0 \le c_i \le 10^9$) — the number of coins each knight has.

Output

Print n integers — the maximum number of coins each knight can have it only he kills other knights.

Examples

input

Copy

```
4 2
4 5 9 7
1 2 11 33
```

output

Copy

```
1 3 46 36
```

input

Copy

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

output

Copy

```
1 3 5 7 9
```

input

Copy

```
1 0
```

```
2
3
```

output

Copy

```
3
```

Note

Consider the first example.

- The first knight is the weakest, so he can't kill anyone. That leaves him with the only coin he initially has.
- The second knight can kill the first knight and add his coin to his own two.
- The third knight is the strongest, but he can't kill more than $k=2$ other knights. It is optimal to kill the second and the fourth knights: $2+11+33=46$.
- The fourth knight should kill the first and the second knights: $33+1+2=36$.

In the second example the first knight can't kill anyone, while all the others should kill the one with the index less by one than their own.

In the third example there is only one knight, so he can't kill anyone.

B. Suit and Tie

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

Allen is hosting a formal dinner party. $2n$ people come to the event in n pairs (couples). After a night of fun, Allen wants to line everyone up for a final picture. The $2n$ people line up, but Allen doesn't like the ordering. Allen prefers if each pair occupies adjacent positions in the line, as this makes the picture more aesthetic.

Help Allen find the minimum number of swaps of **adjacent** positions he must perform to make it so that each couple occupies adjacent positions in the line.

Input

The first line contains a single integer n ($1 \le n \le 100$), the number of pairs of people.

The second line contains $2n$ integers $a_1, a_2, \ldots, a_{2n}$. For each i with $1 \le i \le n$, i appears exactly twice. If $a_j = a_k = i$, that means that the j-th and k-th people in the line form a couple.

Output

Output a single integer, representing the minimum number of adjacent swaps needed to line the people up so that each pair occupies adjacent positions.

Examples

input

Copy

```
4
```

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

output

Copy

```
2
```

input

Copy

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

output

Copy

```
0
```

input

Copy

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

output

Copy

```
3
```

Note

In the first sample case, we can transform $11233244 \rightarrow 11232344 \rightarrow 11223344$ in two steps. Note that the sequence $11233244 \rightarrow 11323244 \rightarrow 11332244$ also works in the same number of steps.

The second sample case already satisfies the constraints; therefore we need 0 swaps.

B. World Cup

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

Allen wants to enter a fan zone that occupies a round square and has n entrances.

There already is a queue of a_i people in front of the i-th entrance. Each entrance allows one person from its queue to enter the fan zone in one minute.

Allen uses the following strategy to enter the fan zone:

- Initially he stands in the end of the queue in front of the first entrance.

- Each minute, if he is not allowed into the fan zone during the minute (meaning he is not the first in the queue), he leaves the current queue and stands in the end of the queue of the next entrance (or the first entrance if he leaves the last entrance).

Determine the entrance through which Allen will finally enter the fan zone.

Input

The first line contains a single integer n ($2 \le n \le 10^5$) — the number of entrances.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$) — the number of people in queues. These numbers do not include Allen.

Output

Print a single integer — the number of entrance that Allen will use.

Examples

input

Copy

```
4
2 3 2 0
```

output

Copy

```
3
```

input

Copy

```
2
10 10
```

output

Copy

```
1
```

input

Copy

```
6
5 2 6 5 7 4
```

output

Copy

```
6
```

Note

In the first example the number of people (not including Allen) changes as follows: $[2,3,2,0] \to [1,2,1,0] \to [0,1,0,0]$. The number in bold is the queue Alles stands in. We see that he will enter the fan zone through the third entrance.

In the second example the number of people (not including Allen) changes as follows: $[10,10] \to [9,9] \to [8,8] \to [7,7] \to [6,6] \to [5,5] \to [4,4] \to [3,3] \to [2,2] \to [1,1] \to [0,0]$.

In the third example the number of people (not including Allen) changes as follows: $[5,2,6,5,7,4] \to [4,1,5,4,6,3] \to [3,0,4,3,5,2] \to [2,0,3,2,4,1] \to [1,0,2,1,3,0] \to [0,0,1,0,2,0]$

B. Binary String Constructing

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

You are given three integers a, b and x. Your task is to construct a binary string s of length $n=a+b$ such that there are exactly a zeroes, exactly b ones and exactly x indices i (where $1 \le i < n$) such that $s_i \ne s_{i+1}$. It is guaranteed that the answer always exists.

For example, for the string "01010" there are four indices i such that $1 \le i < n$ and $s_i \ne s_{i+1}$ ($i=1,2,3,4$). For the string "111001" there are two such indices i ($i=3,5$).

Recall that binary string is a non-empty sequence of characters where each character is either 0 or 1.

Input

The first line of the input contains three integers a, b and x ($1 \le a,b \le 100, 1 \le x < a+b$).

Output

Print only one string s, where s is **any** binary string satisfying conditions described above. It is guaranteed that the answer always exists.

Examples

input

Copy

```
2 2 1
```

output

Copy

```
1100
```

input

Copy

```
3 3 3
```

output

Copy

```
101100
```

input

Copy

```
5 3 6
```

output

Copy

```
01010100
```

Note

All possible answers for the first example:

- `1100`;
- `0011`.

All possible answers for the second example:

- `110100`;
- `101100`;
- `110010`;
- `100110`;
- `011001`;
- `001101`;
- `010011`;
- `001011`.

B. Sonya and Exhibition

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

Sonya decided to organize an exhibition of flowers. Since the girl likes only roses and lilies, she decided that only these two kinds of flowers should be in this exhibition.

There are n flowers in a row in the exhibition. Sonya can put either a rose or a lily in the i-th position. Thus each of n positions should contain exactly one flower: a rose or a lily.

She knows that exactly m people will visit this exhibition. The i-th visitor will visit all flowers from l_i to r_i inclusive. The girl knows that each segment has its own *beauty* that is equal to the product of the number of roses and the number of lilies.

Sonya wants her exhibition to be liked by a lot of people. That is why she wants to put the flowers in such way that the sum of *beauties* of all segments would be maximum possible.

Input

The first line contains two integers n and m ($1 \le n, m \le 10^3$) — the number of flowers and visitors respectively.

Each of the next m lines contains two integers l_i and r_i ($1 \le l_i \le r_i \le n$), meaning that i-th visitor will visit all flowers from l_i to r_i inclusive.

Output

Print the string of n characters. The i-th symbol should be «`0`» if you want to put a rose in the i-th position, otherwise «`1`» if you want to put a lily.

If there are multiple answers, print any.

Examples

input

Copy

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

output

Copy

```
01100
```

input

Copy

```
6 3
5 6
1 4
4 6
```

output

Copy

```
110010
```

Note

In the first example, Sonya can put roses in the first, fourth, and fifth positions, and lilies in the second and third positions;

- in the segment $[1\ldots3]$, there are one rose and two lilies, so the *beauty* is equal to $1\cdot2=2$;
- in the segment $[2\ldots4]$, there are one rose and two lilies, so the *beauty* is equal to $1\cdot2=2$;
- in the segment $[2\ldots5]$, there are two roses and two lilies, so the *beauty* is equal to $2\cdot2=4$.

The total *beauty* is equal to $2+2+4=8$.

In the second example, Sonya can put roses in the third, fourth, and sixth positions, and lilies in the first, second, and fifth positions;

- in the segment $[5\ldots6]$, there are one rose and one lily, so the *beauty* is equal to $1\cdot1=1$;
- in the segment $[1\ldots4]$, there are two roses and two lilies, so the *beauty* is equal to $2\cdot2=4$;
- in the segment $[4\ldots6]$, there are two roses and one lily, so the *beauty* is equal to $2\cdot1=2$.

The total *beauty* is equal to $1+4+2=7$.

B. Minimum Ternary String

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

You are given a ternary string (it is a string which consists only of characters '0', '1' and '2').

You can swap any two adjacent (consecutive) characters '0' and '1' (i.e. replace "01" with "10" or vice versa) or any two adjacent (consecutive) characters '1' and '2' (i.e. replace "12" with "21" or vice versa).

For example, for string "010210" we can perform the following moves:

- "010210" → "100210";
- "010210" → "001210";
- "010210" → "010120";
- "010210" → "010201".

Note than you cannot swap "02" → "20" and vice versa. You cannot perform any other operations with the given string excluding described above.

You task is to obtain the minimum possible (lexicographically) string by using these swaps arbitrary number of times (*possibly, zero*).

String a is lexicographically less than string b (if strings a and b have the same length) if there exists some position i ($1 \le i \le |a|$, where $|s|$ is the length of the string s) such that for every $j < i$ holds $a_j = b_j$, and $a_i < b_i$.

Input

The first line of the input contains the string s consisting only of characters '0', '1' and '2', its length is between 1 and 10^5 (inclusive).

Output

Print a single string — the minimum possible (lexicographically) string you can obtain by using the swaps described above arbitrary number of times (*possibly, zero*).

Examples

input

Copy

```
100210
```

output

Copy

```
001120
```

input

Copy

```
11222121
```

output

Copy

```
11112222
```

input

Copy

```
20
```

output

Copy

```
20
```

B. Segment Occurrences

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

You are given two strings s and t, both consisting only of lowercase Latin letters.

The substring $s[l..r]$ is the string which is obtained by taking characters $s_l, s_{l+1}, \ldots, s_r$ without changing the order.

Each of the occurrences of string a in a string b is a position i ($1 \le i \le |b| - |a| + 1$) such that $b[i..i+|a|-1] = a$ ($|a|$ is the length of string a).

You are asked q queries: for the i-th query you are required to calculate the number of occurrences of string t in a substring $s[l_i..r_i]$.

Input

The first line contains three integer numbers n, m and q ($1 \le n, m \le 10^3$, $1 \le q \le 10^5$) — the length of string s, the length of string t and the number of queries, respectively.

The second line is a string s ($|s| = n$), consisting only of lowercase Latin letters.

The third line is a string t ($|t| = m$), consisting only of lowercase Latin letters.

Each of the next q lines contains two integer numbers l_i and r_i ($1 \le l_i \le r_i \le n$) — the arguments for the i-th query.

Output

Print q lines — the i-th line should contain the answer to the i-th query, that is the number of occurrences of string t in a substring $s[l_i..r_i]$.

Examples

input

Copy

```
10 3 4
codeforces
for
1 3
3 10
5 6
5 7
```

output

Copy

```
0
1
0
1
```

input

Copy

```
15 2 3
```

```
abacabadabacaba
ba
1 15
3 4
2 14
```

output

Copy

```
4
0
3
```

input

Copy

```
3 5 2
aaa
baaab
1 3
1 1
```

output

Copy

```
0
0
```

Note

In the first example the queries are substrings: "cod", "deforces", "fo" and "for", respectively.

B. Shashlik Cooking

time limit per test
1 second
memory limit per test
512 megabytes
input
standard input
output
standard output

Long story short, shashlik is Miroslav's favorite food. Shashlik is prepared on several skewers simultaneously. There are two states for each skewer: initial and turned over.

This time Miroslav laid out n skewers parallel to each other, and enumerated them with consecutive integers from 1 to n in order from left to right. For better cooking, he puts them quite close to each other, so when he turns skewer number i, it leads to turning k closest skewers from each side of the skewer i, that is, skewers number $i-k$, $i-k+1$, ..., $i-1$, $i+1$, ..., $i+k-1$, $i+k$ (if they exist).

For example, let $n=6$ and $k=1$. When Miroslav turns skewer number 3, then skewers with numbers 2, 3, and 4 will come up turned over. If after that he turns skewer number 1,

then skewers number 1, 3, and 4 will be turned over, while skewer number 2 will be in the initial position (because it is turned again).

As we said before, the art of cooking requires perfect timing, so Miroslav wants to turn over all n skewers with the minimal possible number of actions. For example, for the above example $n=6$ and $k=1$, two turnings are sufficient: he can turn over skewers number 2 and 5.

Help Miroslav turn over all n skewers.

Input

The first line contains two integers n and k ($1 \le n \le 1000$, $0 \le k \le 1000$) — the number of skewers and the number of skewers from each side that are turned in one step.

Output

The first line should contain integer l — the minimum number of actions needed by Miroslav to turn over all n skewers. After than print l integers from 1 to n denoting the number of the skewer that is to be turned over at the corresponding step.

Examples

input

Copy

```
7 2
```

output

Copy

```
2
1 6
```

input

Copy

```
5 1
```

output

Copy

```
2
1 4
```

Note

In the first example the first operation turns over skewers 1, 2 and 3, the second operation turns over skewers 4, 5, 6 and 7.

In the second example it is also correct to turn over skewers 2 and 5, but turning skewers 2 and 4, or 1 and 5 are incorrect solutions because the skewer 3 is in the initial state after these operations.

B. Alice and Hairdresser

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

output
standard output

Alice's hair is growing by leaps and bounds. Maybe the cause of it is the excess of vitamins, or maybe it is some black magic...

To prevent this, Alice decided to go to the hairdresser. She wants for her hair length to be at most l centimeters after haircut, where l is her favorite number. Suppose, that the Alice's head is a straight line on which n hairlines grow. Let's number them from 1 to n. With one swing of the scissors the hairdresser can shorten all hairlines on any segment to the length l, given that **all** hairlines on that segment had length **strictly greater** than l. The hairdresser wants to complete his job as fast as possible, so he will make the least possible number of swings of scissors, since each swing of scissors takes one second.

Alice hasn't decided yet when she would go to the hairdresser, so she asked you to calculate how much time the haircut would take depending on the time she would go to the hairdresser. In particular, you need to process queries of two types:

- 0 — Alice asks how much time the haircut would take if she would go to the hairdresser now.
- $1\ p\ d$ — p-th hairline grows by d centimeters.

Note, that in the request 0 Alice is interested in hypothetical scenario of taking a haircut now, so no hairlines change their length.

Input

The first line contains three integers n, m and l ($1 \le n, m \le 100000$, $1 \le l \le 10^9$) — the number of hairlines, the number of requests and the favorite number of Alice.

The second line contains n integers a_i ($1 \le a_i \le 10^9$) — the initial lengths of all hairlines of Alice.

Each of the following m lines contains a request in the format described in the statement.

The request description starts with an integer t_i. If $t_i = 0$, then you need to find the time the haircut would take. Otherwise, $t_i = 1$ and in this moment one hairline grows. The rest of the line than contains two more integers: p_i and d_i ($1 \le p_i \le n$, $1 \le d_i \le 10^9$) — the number of the hairline and the length it grows by.

Output

For each query of type 0 print the time the haircut would take.

Example

input

Copy

```
4 7 3
1 2 3 4
0
1 2 3
0
1 1 3
0
1 3 1
0
```

output

Copy

```
1
2
```

```
2
1
```

Note

Consider the first example:

- Initially lengths of hairlines are equal to $1,2,3,4$ and only 4-th hairline is longer $l=3$, and hairdresser can cut it in 1 second.
- Then Alice's second hairline grows, the lengths of hairlines are now equal to $1,5,3,4$
- Now haircut takes two seonds: two swings are required: for the 4-th hairline and for the 2-nd.
- Then Alice's first hairline grows, the lengths of hairlines are now equal to $4,5,3,4$
- The haircut still takes two seconds: with one swing hairdresser can cut 4-th hairline and with one more swing cut the segment from 1-st to 2-nd hairline.
- Then Alice's third hairline grows, the lengths of hairlines are now equal to $4,5,4,4$
- Now haircut takes only one second: with one swing it is possible to cut the segment from 1-st hairline to the 4-th.

B. DDoS

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

We get more and more news about DDoS-attacks of popular websites.
Arseny is an admin and he thinks that a website is under a DDoS-attack if the total number of requests for a some period of time exceeds $100 \cdot t$, where t — the number of seconds in this time segment.
Arseny knows statistics on the number of requests per second since the server is booted. He knows the sequence $r_1, r_2, \ldots, r_n$, where r_i — the number of requests in the i-th second after boot.
Determine the length of the longest continuous period of time, which Arseny considers to be a DDoS-attack. A seeking time period should not go beyond the boundaries of the segment $[1, n]$.

Input

The first line contains n ($1 \le n \le 5000$) — number of seconds since server has been booted. The second line contains sequence of integers $r_1, r_2, \ldots, r_n$ ($0 \le r_i \le 5000$), r_i — number of requests in the i-th second.

Output

Print the only integer number — the length of the longest time period which is considered to be a DDoS-attack by Arseny. If it doesn't exist print `0`.

Examples

input

Copy

```
5
100 200 1 1 1
```

output

Copy

```
3
```

input

Copy

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

output

Copy

```
0
```

input

Copy

```
2
101 99
```

output

Copy

```
1
```

B. Forgery

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

Student Andrey has been skipping physical education lessons for the whole term, and now he must somehow get a passing grade on this subject. Obviously, it is impossible to do this by legal means, but Andrey doesn't give up. Having obtained an empty certificate from a local hospital, he is going to use his knowledge of local doctor's handwriting to make a counterfeit certificate of illness. However, after writing most of the certificate, Andrey suddenly discovered that doctor's signature is impossible to forge. Or is it?

For simplicity, the signature is represented as an $n \times m$ grid, where every cell is either filled with ink or empty. Andrey's pen can fill a 3×3 square without its central cell if it is completely contained inside the grid, as shown below.

```
xxx
x.x
xxx
```

Determine whether is it possible to forge the signature on an empty $n \times m$ grid.

Input

The first line of input contains two integers n and m ($3 \le n, m \le 1000$).

Then n lines follow, each contains m characters. Each of the characters is either '.', representing an empty cell, or '#', representing an ink filled cell.

Output

If Andrey can forge the signature, output "YES". Otherwise output "NO".

You can print each letter in any case (upper or lower).

Examples

input

Copy

```
3 3
###
#.#
###
```

output

Copy

```
YES
```

input

Copy

```
3 3
###
###
###
```

output

Copy

```
NO
```

input

Copy

```
4 3
###
###
###
###
```

output

Copy

```
YES
```

input

Copy

```
5 7
.......
.#####.
.#.#.#.
.#####.
.......
```

output

Copy

```
YES
```

Note

In the first sample Andrey can paint the border of the square with the center in $(2,2)$.

In the second sample the signature is impossible to forge.

In the third sample Andrey can paint the borders of the squares with the centers in $(2,2)$ and $(3,2)$:

1. we have a clear paper:

```
...
...
...
...
```

   - use the pen with center at $(2,2)$.

```
###
#.#
###
...
```

   - use the pen with center at $(3,2)$.

```
###
###
###
###
```

In the fourth sample Andrey can paint the borders of the squares with the centers in $(3,3)$ and $(3,5)$.

B. Vasya and Isolated Vertices

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

Vasya has got an undirected graph consisting of n vertices and m edges. This graph doesn't contain any self-loops or multiple edges. Self-loop is an edge connecting a vertex to itself. Multiple edges are a pair of edges such that they connect the same pair of vertices. Since the graph is undirected, the pair of edges $(1,2)$ and $(2,1)$ is considered to

be multiple edges. Isolated vertex of the graph is a vertex such that there is no edge connecting this vertex to any other vertex.
Vasya wants to know the minimum and maximum possible number of isolated vertices in an undirected graph consisting of n vertices and m edges.

Input

The only line contains two integers n and m ($1 \le n \le 10^5, 0 \le m \le \frac{n(n-1)}{2}$).
It is guaranteed that there exists a graph without any self-loops or multiple edges with such number of vertices and edges.

Output

In the only line print two numbers min and max — the minimum and maximum number of isolated vertices, respectively.

Examples

input

Copy

```
4 2
```

output

Copy

```
0 1
```

input

Copy

```
3 1
```

output

Copy

```
1 1
```

Note

In the first example it is possible to construct a graph with 0 isolated vertices: for example, it should contain edges $(1,2)$ and $(3,4)$. To get one isolated vertex, we may construct a graph with edges $(1,2)$ and $(1,3)$.
In the second example the graph will always contain exactly one isolated vertex.

B. Build a Contest

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

Arkady coordinates rounds on some not really famous competitive programming platform. Each round features n problems of distinct difficulty, the difficulties are numbered from 1 to n.

To hold a round Arkady needs n new (not used previously) problems, one for each difficulty. As for now, Arkady creates all the problems himself, but unfortunately, he can't just create a problem of a desired difficulty. Instead, when he creates a problem, he evaluates its difficulty from 1 to n and puts it into the problems pool.

At each moment when Arkady can choose a set of n new problems of distinct difficulties from the pool, he holds a round with these problems and removes them from the pool. Arkady always creates one problem at a time, so if he can hold a round after creating a problem, he immediately does it.

You are given a sequence of problems' difficulties in the order Arkady created them. For each problem, determine whether Arkady held the round right after creating this problem, or not. Initially the problems pool is empty.

Input

The first line contains two integers n and m ($1 \le n, m \le 10^5$) — the number of difficulty levels and the number of problems Arkady created.

The second line contains m integers $a_1, a_2, \ldots, a_m$ ($1 \le a_i \le n$) — the problems' difficulties in the order Arkady created them.

Output

Print a line containing m digits. The i-th digit should be 1 if Arkady held the round after creation of the i-th problem, and 0 otherwise.

Examples

input

Copy

```
3 11
2 3 1 2 2 2 3 2 2 3 1
```

output

Copy

```
00100000001
```

input

Copy

```
4 8
4 1 3 3 2 3 3 3
```

output

Copy

```
00001000
```

Note

In the first example Arkady held the round after the first three problems, because they are of distinct difficulties, and then only after the last problem

B. Accordion

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

An *accordion* is a string (yes, in the real world accordions are musical instruments, but let's forget about it for a while) which can be represented as a concatenation of: an opening bracket (ASCII code 091), a colon (ASCII code 058), some (possibly zero) vertical line characters (ASCII code 124), another colon, and a closing bracket (ASCII code 093). The length of the accordion is the number of characters in it.
For example, `[::]`, `[:||:]` and `[:|||:]` are accordions having length 4, 6 and 7. `(:|:)`, `{:||:}`, `[:]`, `]:||:[` are not accordions.

You are given a string s. You want to transform it into an accordion by removing some (possibly zero) characters from it. Note that you may not insert new characters or reorder existing ones. Is it possible to obtain an accordion by removing characters from s, and if so, what is the maximum possible length of the result?

Input

The only line contains one string s ($1 \le |s| \le 500000$). It consists of lowercase Latin letters and characters `[`, `]`, `:` and `|`.

Output

If it is not possible to obtain an accordion by removing some characters from s, print -1. Otherwise print maximum possible length of the resulting accordion.

Examples

input

Copy

```
|[a:b:|]
```

output

Copy

```
4
```

input

Copy

```
|]:[|:]
```

output

Copy

```
-1
```

B. Array K-Coloring

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

You are given an array a consisting of n integer numbers.

You have to color this array in k colors in such a way that:

- Each element of the array should be colored in some color;
- For each i from 1 to k there should be **at least one** element colored in the i-th color in the array;
- For each i from 1 to k all elements colored in the i-th color should be **distinct**.

Obviously, such coloring might be impossible. In this case, print "NO". Otherwise print "YES" and **any** coloring (i.e. numbers $c_1,c_2,\ldots c_n$, where $1 \le c_i \le k$ and c_i is the color of the i-th element of the given array) satisfying the conditions above. If there are multiple answers, you can print **any**.

Input

The first line of the input contains two integers n and k ($1 \le k \le n \le 5000$) — the length of the array a and the number of colors, respectively.

The second line of the input contains n integers $a_1,a_2,\ldots,a_n$ ($1 \le a_i \le 5000$) — elements of the array a.

Output

If there is no answer, print "NO". Otherwise print "YES" and **any** coloring (i.e. numbers $c_1,c_2,\ldots c_n$, where $1 \le c_i \le k$ and c_i is the color of the i-th element of the given array) satisfying the conditions described in the problem statement. If there are multiple answers, you can print **any**.

Examples

input

Copy

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

output

Copy

```
YES
1 1 2 2
```

input

Copy

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

output

Copy

```
YES
```

```
2 1 1 2 1
```

input

Copy

```
5 2
2 1 1 2 1
```

output

Copy

```
NO
```

Note

In the first example the answer $2\ 1\ 2\ 1$ is also acceptable.

In the second example the answer $1\ 1\ 1\ 2\ 2$ is also acceptable.

There exist other acceptable answers for both examples.

B. Tape

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

You have a long stick, consisting of m segments enumerated from 1 to m. Each segment is 1 centimeter long. Sadly, some segments are broken and need to be repaired.

You have an infinitely long repair tape. You want to cut some pieces from the tape and use them to cover all of the broken segments. To be precise, a piece of tape of integer length t placed at some position s will cover segments $s,s+1,\ldots,s+t-1$.

You are allowed to cover non-broken segments; it is also possible that some pieces of tape will overlap.

Time is money, so you want to cut at most k continuous pieces of tape to cover all the broken segments. What is the minimum total length of these pieces?

Input

The first line contains three integers n, m and k ($1 \le n \le 10^5$, $n \le m \le 10^9$, $1 \le k \le n$) — the number of broken segments, the length of the stick and the maximum number of pieces you can use.

The second line contains n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le m$) — the positions of the broken segments. These integers are given in increasing order, that is, $b_1 < b_2 < \ldots < b_n$.

Output

Print the minimum total length of the pieces.

Examples

input

Copy

```
4 100 2
20 30 75 80
```

output

Copy

```
17
```

input

Copy

```
5 100 3
1 2 4 60 87
```

output

Copy

```
6
```

Note

In the first example, you can use a piece of length 11 to cover the broken segments 20 and 30, and another piece of length 6 to cover 75 and 80, for a total length of 17.

In the second example, you can use a piece of length 4 to cover broken segments 1, 2 and 4, and two pieces of length 1 to cover broken segments 60 and 87.

B. Sasha and Magnetic Machines

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

One day Sasha visited the farmer 2D and his famous magnetic farm. On this farm, the crop grows due to the influence of a special magnetic field. Maintaining of the magnetic field is provided by n machines, and the power of the i-th machine is a_i.

This year 2D decided to cultivate a new culture, but what exactly he didn't say. For the successful growth of the new culture, it is necessary to slightly change the powers of the machines. 2D can **at most once** choose an arbitrary integer x, then choose one machine and reduce the power of its machine by x times, and at the same time increase the power of one another machine by x times (powers of all the machines must stay **positive integers**). Note that he may not do that if he wants. More formally, 2D can choose two

such indices i and j, and one integer x such that x is a divisor of a_i, and change powers as following: $a_i = \frac{a_i}{x}$, $a_j = a_j \cdot x$

Sasha is very curious, that's why he wants to calculate the **minimum** total power the farmer can reach. There are too many machines, and Sasha can't cope with computations, help him!

Input

The first line contains one integer n ($2 \le n \le 5 \cdot 10^4$) — the number of machines.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 100$) — the powers of the machines.

Output

Print one integer — minimum total power.

Examples

input

Copy

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

output

Copy

```
14
```

input

Copy

```
4
4 2 4 4
```

output

Copy

```
14
```

input

Copy

```
5
2 4 2 3 7
```

output

Copy

```
18
```

Note

In the first example, the farmer can reduce the power of the 4-th machine by 2 times, and increase the power of the 1-st machine by 2 times, then the powers will be: $[2,2,3,2,5]$.

In the second example, the farmer can reduce the power of the 3-rd machine by 2 times, and increase the power of the 2-nd machine by 2 times. At the same time, the farmer can leave is be as it is and the total power won't change.

In the third example, it is optimal to leave it be as it is

B. Draw!

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

You still have partial information about the score during the historic football match. You are given a set of pairs (a_i,b_i), indicating that at some point during the match the score was "a_i: b_i". It is known that if the current score is «x:y», then after the goal it will change to "$x+1$:y" or "x:$y+1$". What is the largest number of times a draw could appear on the scoreboard?

The pairs "a_i:b_i" are given in chronological order (time increases), but you are given score only for some moments of time. The last pair corresponds to the end of the match.

Input

The first line contains a single integer n ($1 \le n \le 10000$) — the number of known moments in the match.

Each of the next n lines contains integers a_i and b_i ($0 \le a_i, b_i \le 10^9$), denoting the score of the match at that moment (that is, the number of goals by the first team and the number of goals by the second team).

All moments are given in chronological order, that is, sequences x_i and y_j are non-decreasing. The last score denotes the final result of the match.

Output

Print the maximum number of moments of time, during which the score was a draw. The starting moment of the match (with a score 0:0) is also counted.

Examples

input

Copy

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

output

Copy

```
2
```

input

Copy

```
3
0 0
0 0
0 0
```

output

Copy

```
1
```

input

Copy

```
1
5 4
```

output

Copy

```
5
```

Note

In the example one of the possible score sequences leading to the maximum number of draws is as follows: 0:0, 1:0, 2:0, 2:1, 3:1, 3:2, 3:3, 3:4.

B. Neko Performs Cat Furrier Transform

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

Cat Furrier Transform is a popular algorithm among cat programmers to create longcats. As one of the greatest cat programmers ever exist, Neko wants to utilize this algorithm to create the perfect longcat.

Assume that we have a cat with a number x. A perfect longcat is a cat with a number equal 2^m-1 for some non-negative integer m. For example, the numbers 0, 1, 3, 7, 15 and so on are suitable for the perfect longcats.

In the Cat Furrier Transform, the following operations can be performed on x:

- (Operation A): you select any non-negative integer n and replace x with $x \oplus (2^n-1)$, with $\oplus$ being a bitwise XOR operator.
- (Operation B): replace x with $x+1$.

The first applied operation must be of type A, the second of type B, the third of type A again, and so on. Formally, if we number operations from one in the order they are executed, then odd-numbered operations must be of type A and the even-numbered operations must be of type B.

Neko wants to produce perfect longcats at industrial scale, thus for each cat Neko only wants to perform at most 40 operations. Can you help Neko writing a transformation plan?

Note that it is **not required** to minimize the number of operations. You just need to use no more than 40 operations.

Input

The only line contains a single integer x ($1 \le x \le 10^6$).

Output

The first line should contain a single integer t ($0 \le t \le 40$) — the number of operations to apply.

Then for each odd-numbered operation print the corresponding number n_i in it. That is, print $\lceil \frac{t}{2} \rceil$ integers n_i ($0 \le n_i \le 30$), denoting the replacement x with $x \oplus (2^{n_i} - 1)$ in the corresponding step.

If there are multiple possible answers, you can print any of them. It is possible to show, that there is at least one answer in the constraints of this problem.

Examples

input

Copy

```
39
```

output

Copy

```
4
5 3
```

input

Copy

```
1
```

output

Copy

```
0
```

input

Copy

```
7
```

output

Copy

```
0
```

Note

In the first test, one of the transforms might be as follows: $39 \to 56 \to 57 \to 62 \to 63$. Or more precisely:

- Pick $n = 5$. x is transformed into $39 \oplus 31$, or 56.
- Increase x by 1, changing its value to 57.
- Pick $n = 3$. x is transformed into $57 \oplus 7$, or 62.
- Increase x by 1, changing its value to $63 = 2^6 - 1$.

In the second and third test, the number already satisfies the goal requirement.

B. Expansion coefficient of the array

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

Let's call an array of non-negative integers $a_1,a_2,\ldots,a_n$ a k-extension for some non-negative integer k if for all possible pairs of indices $1 \le i,j \le n$ the inequality $k \cdot |i-j| \le \min(a_i,a_j)$ is satisfied. The expansion coefficient of the array a is the maximal integer k such that the array a is a k-extension. Any array is a 0-expansion, so the expansion coefficient always exists.

You are given an array of non-negative integers $a_1,a_2,\ldots,a_n$. Find its expansion coefficient.

Input

The first line contains one positive integer n — the number of elements in the array a ($2 \le n \le 300000$). The next line contains n non-negative integers $a_1,a_2,\ldots,a_n$, separated by spaces ($0 \le a_i \le 10^9$).

Output

Print one non-negative integer — expansion coefficient of the array $a_1,a_2,\ldots,a_n$.

Examples

input

Copy

```
4
6 4 5 5
```

output

Copy

```
1
```

input

Copy

```
3
0 1 2
```

output

Copy

```
0
```

input

Copy

```
4
821 500 479 717
```

output

Copy

```
239
```

Note

In the first test, the expansion coefficient of the array $[6,4,5,5]$ is equal to 1 because $|i-j| \le \min(a_i,a_j)$, because all elements of the array satisfy $a_i \ge 3$. On the other hand, this array isn't a 2-extension, because $6 = 2 \cdot |1-4| \le \min(a_1,a_4) = 5$ is false.

In the second test, the expansion coefficient of the array $[0,1,2]$ is equal to 0 because this array is not a 1-extension, but it is 0-extension.

B. Lost Numbers

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

This is an interactive problem. Remember to flush your output while communicating with the testing program. You may use `fflush(stdout)` in C++, `system.out.flush()` in Java, `stdout.flush()` in Python or `flush(output)` in Pascal to flush the output. If you use some other programming language, consult its documentation. You may also refer to the guide on interactive problems: https://codeforces.com/blog/entry/45307.

The jury guessed some array a consisting of 6 integers. There are 6 special numbers — 4, 8, 15, 16, 23, 42 — and each of these numbers occurs in a exactly once (so, a is some permutation of these numbers).

You don't know anything about their order, but you are allowed to ask **up to 4 queries**. In each query, you may choose two indices i and j ($1 \le i,j \le 6$, i and j are not necessarily distinct), and you will get the value of $a_i \cdot a_j$ in return.

Can you guess the array a?

The array a is fixed beforehand in each test, the interaction program doesn't try to adapt to your queries.

Interaction

Before submitting the answer, you may ask up to 4 queries. To ask a query, print one line in the following format: ? i j, where i and j should be two integers such that $1 \le i,j \le 6$. *The line should be ended with a line break character*. After submitting a query, flush the output and read the answer to your query — one line containing one integer $a_i \cdot a_j$. If you submit an incorrect query (or ask more than 4 queries), the answer to it will be one string `0`. After receiving such an answer, your program should terminate immediately — otherwise you may receive verdict "Runtime error", "Time limit exceeded" or some other verdict instead of "Wrong answer".

To give the answer, your program should print one line ! a_1 a_2 a_3 a_4 a_5 a_6 *with a line break in the end*. After that, it should flush the output and terminate gracefully.

Example

input

Copy

```
16
64
345
672
```

output

Copy

```
? 1 1
? 2 2
? 3 5
? 4 6
! 4 8 15 16 23 42
```

Note

If you want to submit a hack for this problem, your test should contain exactly six space-separated integers a_1, a_2, ..., a_6. Each of 6 special numbers should occur exactly once in the test. The test should be ended with a line break character.