Codeforces Round #655 (Div. 2)

A. Omkar and Completion

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

You have been blessed as a child of Omkar. To express your gratitude, please solve this problem for Omkar!

An array a of length n is called **complete** if all elements are positive and don't exceed 1000, and for all indices x,y,z ($1 \leq x, y, z \leq n$), $a_x + a_y \neq a_z$ (not necessarily distinct).

You are given one integer n. Please find any **complete** array of length n. It is guaranteed that under given constraints such array exists.

Input

Each test contains multiple test cases. The first line contains t ($1 \le t \le 1000$) — the number of test cases. Description of the test cases follows.

The only line of each test case contains one integer n ($1 \leq n \leq 1000$).

It is guaranteed that the sum of n over all test cases does not exceed 1000.

Output

For each test case, print a complete array on a single line. All elements have to be integers between 1 and 1000 and for all indices x, y,z ($1 \leq x, y, z \leq n$) (not necessarily distinct), $a_x + a_y \neq a_z$ must hold.

If multiple solutions exist, you may print any.

Example

input

```
2
5
4
```

output

```
1 5 3 77 12
384 384 44 44
```

Note

It can be shown that the outputs above are valid for each test case. For example, $44+44 \ne 384$.

Below are some examples of arrays that are NOT **complete** for the 1st test case:

$[1,2,3,4,5]$

Notice that $a_{1} + a_{2} = a_{3}$.

$[1,3000,1,300,1]$

Notice that $a_2 = 3000 > 1000$.

B. Omkar and Last Class of Math

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

In Omkar's last class of math, he learned about the least common multiple, or LCM. $LCM(a, b)$ is the smallest positive integer x which is divisible by both a and b.

Omkar, having a laudably curious mind, immediately thought of a problem involving the LCM operation: given an integer n, find positive integers a and b such that $a + b = n$ and $LCM(a, b)$ is the minimum value possible.

Can you help Omkar solve his ludicrously challenging math problem?

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 10$). Description of the test cases follows.

Each test case consists of a single integer n ($2 \le n \le 10^9$).

Output

For each test case, output two positive integers a and b, such that $a + b = n$ and $LCM(a, b)$ is the minimum possible.

Example

input

```
3
4
6
9
```

output

```
2 2
3 3
3 6
```

Note

For the first test case, the numbers we can choose are $1, 3$ or $2, 2$. $LCM(1, 3) = 3$ and $LCM(2, 2) = 2$, so we output $2\ 2$.

For the second test case, the numbers we can choose are $1, 5$, $2, 4$, or $3, 3$. $LCM(1, 5) = 5$, $LCM(2, 4) = 4$, and $LCM(3, 3) = 3$, so we output $3\ 3$.

For the third test case, $LCM(3, 6) = 6$. It can be shown that there are no other pairs of numbers which sum to 9 that have a lower LCM.

C. Omkar and Baseball

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

Patrick likes to play baseball, but sometimes he will spend so many hours hitting home runs that his mind starts to get foggy! Patrick is sure that his scores across n sessions follow the identity permutation (ie. in the first game he scores 1 point, in the second game he scores 2 points and so on). However, when he checks back to his record, he sees that all the numbers are mixed up!

Define a special exchange as the following: choose any subarray of the scores and permute elements such that no element of subarray gets to the same position as it was before the exchange. For example, performing a special exchange on $[1, 2, 3]$ can yield $[3, 1, 2]$ but it cannot yield $[3, 2, 1]$ since the 2 is in the same position.

Given a permutation of n integers, please help Patrick find the minimum number of special exchanges needed to make the permutation sorted! It can be proved that under given constraints this number doesn't exceed 10^{18}.

An array a is a subarray of an array b if a can be obtained from b by deletion of several (possibly, zero or all) elements from the beginning and several (possibly, zero or all) elements from the end.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 100$). Description of the test cases follows.

The first line of each test case contains integer n ($1 \le n \le 2 \cdot 10^5$) — the length of the given permutation.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the initial permutation.

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

Output

For each test case, output one integer: the minimum number of special exchanges needed to sort the permutation.

Example

input

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

output

```
0
2
```

Note

In the first permutation, it is already sorted so no exchanges are needed.

It can be shown that you need at least 2 exchanges to sort the second permutation.

$[3, 2, 4, 5, 1, 6, 7]$

Perform special exchange on range $(1, 5)$

$[4, 1, 2, 3, 5, 6, 7]$

Perform special exchange on range $(1, 4)$

$[1, 2, 3, 4, 5, 6, 7]$

D. Omkar and Circle

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

Danny, the local Math Maniac, is fascinated by circles, Omkar's most recent creation. Help him solve this circle problem!

You are given n nonnegative integers $a_1, a_2, \dots, a_n$ arranged in a circle, where n must be odd (ie. $n - 1$ is divisible by 2). Formally, for all i such that $2 \leq i \leq n$, the elements a_{i-1} and a_i are considered to be adjacent, and a_n and a_1 are also considered to be adjacent. In one operation, you pick a number on the circle, replace it with the sum of the two elements adjacent to it, and then delete the two adjacent elements from the circle. This is repeated until only one number remains in the circle, which we call the circular value.

Help Danny find the maximum possible circular value after some sequences of operations.

Input

The first line contains one odd integer n ($1 \leq n < 2 \cdot 10^5$, n is odd) — the initial size of the circle.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($0 \leq a_i \leq 10^9$) — the initial numbers in the circle.

Output

Output the maximum possible circular value after applying some sequence of operations to the given circle.

Examples

input

```
3
7 10 2
```

output

```
17
```

input

```
1
4
```

output

```
4
```

Note

For the first test case, here's how a circular value of 17 is obtained:

Pick the number at index 3. The sum of adjacent elements equals 17. Delete 7 and 10 from the circle and replace 2 with 17.

Note that the answer may not fit in a 32-bit integer.

E. Omkar and Last Floor

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

Omkar is building a house. He wants to decide how to make the floor plan for the last floor.

Omkar's floor starts out as n rows of m zeros ($1 \leq n, m \leq 100$). Every row is divided into intervals such that every 0 in the row is in exactly 1 interval. For every interval for every row, Omkar can change exactly one of the 0s contained in that interval to a 1. Omkar defines the quality of a floor as the sum of the squares of the sums of the values in each column, i. e. if the sum of the values in the i-th column is q_i, then the quality of the floor is $\sum_{i=1}^{m} q_i^2$.

Help Omkar find the maximum quality that the floor can have.

Input

The first line contains two integers, n and m ($1 \leq n, m \leq 100$), which are the number of rows and number of columns, respectively.

You will then receive a description of the intervals in each row. For every row i from 1 to n: The first row contains a single integer k_i ($1 \leq k_i \leq m$), which is the number of intervals on row i. The j-th of the next k_i lines contains two integers $l_{i,j}$ and $r_{i,j}$, which are the left and right bound (both inclusive), respectively, of the j-th interval of the i-th row. It is guaranteed that all intervals other than the first interval will be directly after the interval before it. Formally, $l_{i,1} = 1$, $l_{i,j} \leq r_{i,j}$ for all $1 \leq j \leq k_i$, $r_{i,j-1} + 1 = l_{i,j}$ for all $2 \leq j \leq k_i$, and $r_{i,k_i} = m$.

Output

Output one integer, which is the maximum possible quality of an eligible floor plan.

Example

input

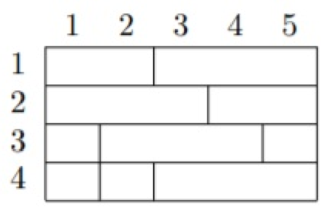
```
4 5
2
1 2
3 5
2
1 3
4 5
3
1 1
2 4
5 5
3
1 1
2 2
3 5
```

output

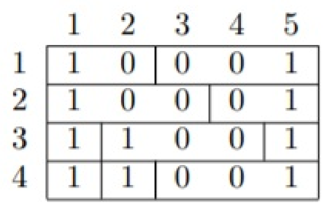
```
36
```

Note

The given test case corresponds to the following diagram. Cells in the same row and have the same number are a part of the same interval.



The most optimal assignment is:



The sum of the 1st column is 4, the sum of the 2nd column is 2, the sum of the 3rd and 4th columns are 0, and the sum of the 5th column is 4.

The quality of this floor plan is $4^2 + 2^2 + 0^2 + 0^2 + 4^2 = 36$. You can show that there is no floor plan with a higher quality.

F. Omkar and Modes

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

Ray lost his array and needs to find it by asking Omkar. Omkar is willing to disclose that the array has the following qualities:

1. The array has n ($1 \leq n \leq 2 \cdot 10^5$) elements.
2. Every element in the array a_i is an integer in the range $1 \leq a_i \leq 10^9$.
3. The array is sorted in nondecreasing order.

Ray is allowed to send Omkar a series of queries. A query consists of two integers, l and r such that $1 \leq l \leq r \leq n$. Omkar will respond with two integers, x and f. x is the mode of the subarray from index l to index r inclusive. The mode of an array is defined by the number that appears the most frequently. If there are multiple numbers that appear the most number of times, the smallest such number is considered to be the mode. f is the amount of times that x appears in the queried subarray.

The array has k ($1 \leq k \leq \min(25000, n)$) distinct elements. However, due to Ray's sins, Omkar will not tell Ray what k is. Ray is allowed to send at most $4k$ queries.

Help Ray find his lost array.

Input

The only line of the input contains a single integer n ($1 \le n \le 2 \cdot 10^5$), which equals to the length of the array that you are trying to find.

Interaction

The interaction starts with reading n.

Then you can make one type of query:

- "$?\ l\ r$" ($1 \le l \le r \le n$) where l and r are the bounds of the subarray that you wish to query.

The answer to each query will be in the form "$x\ f$" where x is the mode of the subarray and f is the number of times x appears in the subarray.

- x satisfies ($1 \le x \le 10^9$).
- f satisfies ($1 \le f \le r - l + 1$).
- If you make more than $4k$ queries or violate the number range in the query, you will get an output "-1."
- If you terminate after receiving the response "-1", you will get the "`Wrong answer`" verdict. Otherwise you can get an arbitrary verdict because your solution will continue to read from a closed stream.

To output your answer, print:

- "$!\ a_1\ a_2\ \ldots\ a_{n-1}\ a_n$" which is an exclamation point followed by the array with a space between every element.

And quit after that. This query is not counted towards the $4k$ queries limit.

After printing a query do not forget to output end of line and flush the output. Otherwise, you will get `Idleness limit exceeded`. To do this, use:

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

Hack Format

To hack, output 1 integer on the first line, n ($1 \le n \le 2 \cdot 10^5$). On the second line output n integers $a_1, a_2, \ldots, a_{n-1}, a_n$ separated by a space such that there are at most 25000 distinct numbers and $a_j \le a_{j+1}$ for all j from 1 to $n-1$.

Example

input

```
6

2 2

2 2

3 2

2 1
```

output

```
? 1 6

? 1 3

? 4 6

? 3 4

! 1 2 2 3 3 4
```

Note

The first query is $l = 1$ and $r = 6$. The mode is 2, and 2 appears 2 times, so $x = 2$ and $f = 2$. Note that 3 also appears two times, but 2 is outputted because 2 is smaller.

The second query is $l = 1$ and $r = 3$. The mode is 2 and 2 appears twice in the subarray with indices $[1, 3]$.

The third query is $l = 4$ and $r = 6$. The mode is 3 and 3 appears twice in the subarray with indices $[4, 6]$.

The fourth query is $l = 3$ and $r = 4$. The mode is 2, which appears once in the subarray with indices $[3, 4]$. Note that 3 also appears once in that range, but 2 is smaller than 3.