Statement is not available on English language

B. Balls of Buma

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Balph is learning to play a game called Buma. In this game, he is given a row of colored balls. He has to choose the color of one new ball and the place to insert it (between two balls, or to the left of all the balls, or to the right of all the balls).

When the ball is inserted the following happens repeatedly: if some segment of balls of the same color became longer as a result of a previous action and its length became at least 3, then all the balls of this segment are eliminated.

Consider, for example, a row of balls 'AAABBBWWBB'. Suppose Balph chooses a ball of color 'W' and the place to insert it after the sixth ball, i. e. to the left of the two 'W's. After Balph inserts this ball, the balls of color 'W' are eliminated, since this segment was made longer and has length 3 now, so the row becomes 'AAABBBBB'. The balls of color 'B' are eliminated now, because the segment of balls of color 'B' became longer and has length 5 now. Thus, the row becomes 'AAA'. However, none of the balls are eliminated now, because there is no elongated segment.

Help Balph count the number of possible ways to choose a color of a new ball and a place to insert it that leads to the elimination of all the balls.

Input

The only line contains a non-empty string of uppercase English letters of length at most $3 \cdot 10^5$. Each letter represents a ball with the corresponding color.

Output

Output the number of ways to choose a color and a position of a new ball in order to eliminate all the balls.

Examples

input
```
BBWWBB
```
output
```
3
```

input
```
BWWB
```
output
```
0
```

input
```
BBWBB
```
output
```
0
```

input
```
OOOWWW
```
output
```
0
```

input
```
WWWOOOOOOWWW
```
output
```
7
```

Statement is not available on English language

D. DevOps Best Practices

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Daisy is a senior software engineer at RainyDay, LLC. She has just implemented three new features in their product: the first feature makes their product work, the second one makes their product fast, and the third one makes their product correct. The company encourages at least some testing of new features, so Daisy appointed her intern Demid to write some tests for the new features.

Interestingly enough, these three features pass all the tests on Demid's development server, which has index 1, but might fail the tests on some other servers.

After Demid has completed this task, Daisy appointed you to deploy these three features to all n servers of your company. For every feature f and every server s, Daisy told you whether she wants the feature f to be deployed on the server s. If she wants it to be deployed, it must be done even if the feature f fails the tests on the server s. If she does not want it to be deployed, you may not deploy it there.

Your company has two important instruments for the deployment of new features to servers: Continuous Deployment (CD) and Continuous Testing (CT). CD can be established between several pairs of servers, forming a directed graph. CT can be set up on some set of servers.

If CD is configured from the server s_1 to the server s_2 then every time s_1 receives a new feature f the system starts the following deployment process of f to s_2:

- If the feature f is already deployed on the server s_2, then nothing is done.
- Otherwise, if CT is not set up on the server s_1, then the server s_1 just deploys the feature f to the server s_2 without any testing.
- Otherwise, the server s_1 runs tests for the feature f. If the tests fail on the server s_1, nothing is done. If the tests pass, then the server s_1 deploys the feature f to the server s_2.

You are to configure the CD/CT system, and after that Demid will deploy all three features on his development server. Your CD/CT system must deploy each feature exactly to the set of servers that Daisy wants.

Your company does not have a lot of computing resources, so you can establish CD from one server to another at most 264 times.

Input

The first line contains integer n ($2 \le n \le 256$) — the number of servers in your company.

Next n lines contain three integers each. The j-th integer in the i-th line is 1 if Daisy wants the j-th feature to be deployed to the i-th server, or 0 otherwise.

Next n lines contain three integers each. The j-th integer in the i-th line is 1 if tests pass for the j-th feature on the i-th server, or 0 otherwise.

Demid's development server has index 1. It is guaranteed that Daisy wants all three features to be deployed to the server number 1, and all three features pass their tests on the server number 1.

Output

If it is impossible to configure CD/CT system with CD being set up between at most 264 pairs of servers, then output the single line "`Impossible`".

Otherwise, the first line of the output must contain the line "`Possible`".

Next line must contain n space-separated integers — the configuration of CT. The i-th integer should be 1 if you set up CT on the i-th server, or 0 otherwise.

Next line must contain the integer m ($0 \le m \le 264$) — the number of CD pairs you want to set up.

Each of the next m lines must describe CD configuration, each line with two integers s_i and t_i ($1 \le s_i, t_i \le n$; $s_i \ne t_i$), establishing automated deployment of new features from the server s_i to the server t_i.

Examples

input

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

output

```
Possible
1 1 1
```

```
2
3 2
1 3
```

input

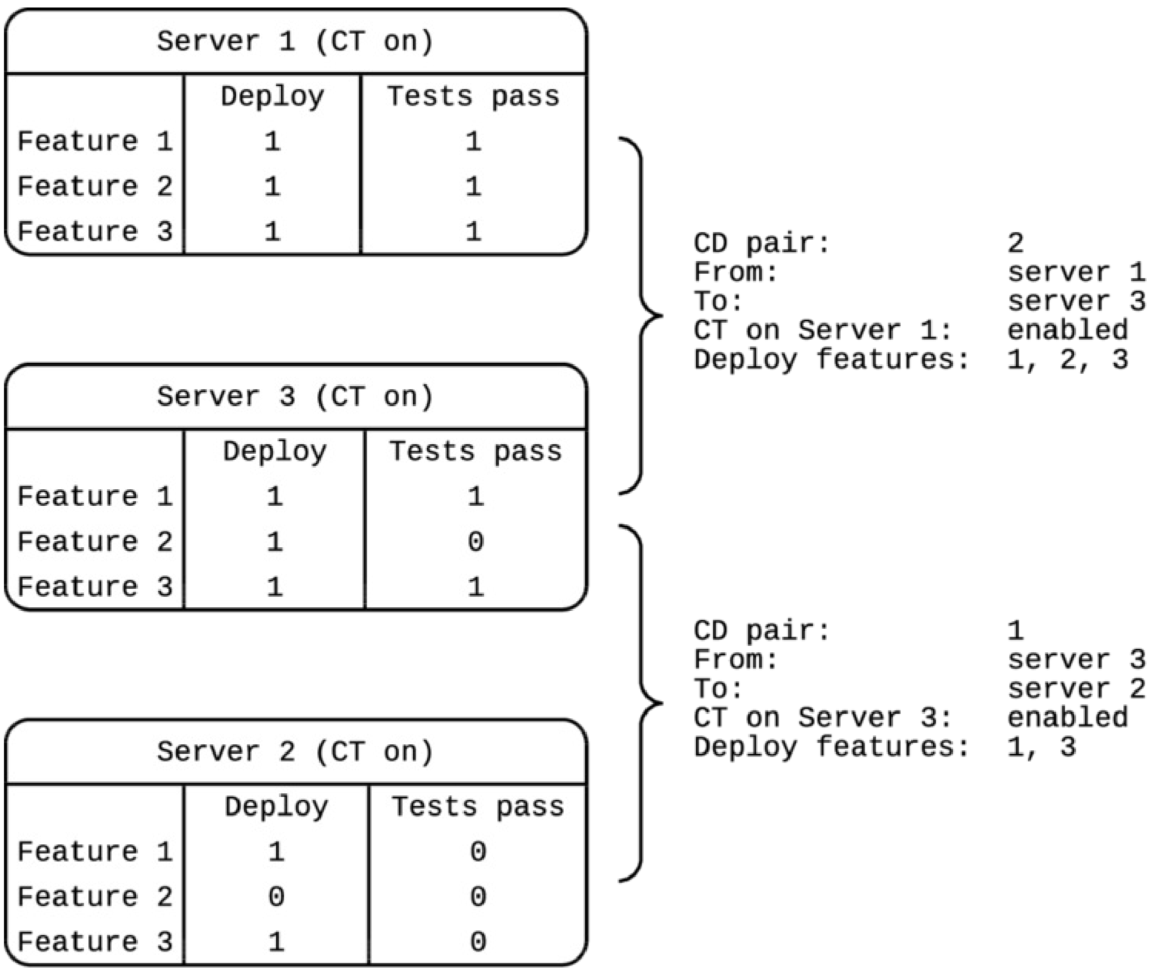
```
2
1 1 1
0 0 1
1 1 1
1 1 0
```

output

```
Impossible
```

Note

CD/CT system for the first sample test is shown below.



Server 1 (CT on)

	Deploy	Tests pass
Feature 1	1	1
Feature 2	1	1
Feature 3	1	1

Server 3 (CT on)

	Deploy	Tests pass
Feature 1	1	1
Feature 2	1	0
Feature 3	1	1

Server 2 (CT on)

	Deploy	Tests pass
Feature 1	1	0
Feature 2	0	0
Feature 3	1	0

```
CD pair:            2
From:               server 1
To:                 server 3
CT on Server 1:     enabled
Deploy features:    1, 2, 3
```

```
CD pair:            1
From:               server 3
To:                 server 2
CT on Server 3:     enabled
Deploy features:    1, 3
```

E. Elections

time limit per test: 3 seconds

memory limit per test: 512 megabytes

input: standard input

output: standard output

Byteburg Senate elections are coming. Usually "United Byteland", the ruling Byteland party, takes all the seats in the Senate to ensure stability and sustainable development. But this year there is one opposition candidate in one of the constituencies. Even one opposition member can disturb the stability in the Senate, so the head of the Party asks you to ensure that the opposition candidate will not be elected.

There are n candidates, numbered from 1 to n. Candidate n is the opposition candidate. There are m polling stations in the constituency, numbered from 1 to m. You know the number of votes cast for each candidate at each polling station. The only thing you can do to prevent the election of the opposition candidate is to cancel the election results at some polling stations. The opposition candidate will be elected if the sum of the votes cast in their favor at all non-canceled stations will be **strictly greater** than the analogous sum for every other candidate.

Your task is to prevent the election of the opposition candidate by canceling the election results at the minimal possible number of polling stations. Notice that solution always exists, because if you cancel the elections at all polling stations, the number of votes for each candidate will be 0, and the opposition candidate will not be elected.

Input

The first line of the input contains two integers n and m ($2 \le n \le 100$; $1 \le m \le 100$) — the number of candidates and the number of polling stations. The next m lines contain the election results at each polling station with n numbers on each line. In the i-th line the j-th number is $a_{i,j}$ — the number of votes cast for the candidate j at the station i ($0 \le a_{i,j} \le 1\,000$).

Output

In the first line output integer k — the minimal number of the polling stations in which you need to cancel the election results. In the second line output k integers — the indices of canceled polling stations, in any order. If there are multiple ways to cancel results at k stations, output any one of them.

Examples

input

```
5 3
6 3 4 2 8
3 7 5 6 7
5 2 4 7 9
```

output

```
2
3 1
```

input

```
2 1
1 1
```

output

```
0
```

input

```
3 3
2 3 8
4 2 9
3 1 7
```

output

```
3
1 2 3
```

Note

In the first example, the candidates from 1 to 5 received 14, 12, 13, 15, and 24 votes correspondingly. The opposition candidate has the most votes. However, if you cancel the election results at the first and the third polling stations, then only the result from the second polling station remains and the vote sums become 3, 7, 5, 6, and 7, without the opposition candidate being in the lead anymore.

Statement is not available on English language

G. Game Relics

time limit per test: 3 seconds

memory limit per test: 512 megabytes

input: standard input

output: standard output

Esports is a form of competitive sports using video games. Dota 2 is one of the most popular competitive video games in Esports. Recently, a new video game Dota 3 was released. In Dota 3 a player can buy some relics for their hero. Relics are counters that track hero's actions and statistics in a game.

Gloria likes to play Dota 3, so she wants to buy all n available relics for her favorite hero.

Relics can be bought using an in-game currency called shards. Each relic has its own price — c_i shards for the i-th relic. A player can buy a relic using one of the following options:

- Pay c_i shards to buy the i-th relic;
- Pay x shards and randomly get one of all n relics. The probability of getting a relic is the same for all n relics. If a duplicate relic is received, then the relic is recycled and $\frac{x}{2}$ shards are given back to the player.

Gloria wants to buy all n relics. Help her minimize the expected number of shards she spends to buy all the relics.

Input

The first line contains two integers n and x ($1 \le n \le 100$; $1 \le x \le 10\,000$) — the number of relics and the cost to receive a random relic.

The second line consists of n integers $c_1, c_2, \ldots, c_n$ ($x \le c_i \le 10\,000$; $\sum c_i \le 10\,000$) — the prices of n relics.

Output

Print a single real number — the minimum expected number of shards that Gloria must spend to buy all the relics.

The absolute or relative error should not exceed 10^{-9}.

Examples

input
```
2 20
25 100
```
output
```
47.50000000000000000
```

input
```
4 30
60 50 60 80
```
output
```
171.25000000000000000
```

Note

In the first example, the optimal strategy is to randomly get one of the two relics paying 20 shards. Then there are two scenarios.

The first one happens if Gloria receives the first relic. Then she keeps getting random relics until she obtains the second relic. The expected number of shards to spend in this scenario is $20 + 30 = 50$.

In the second scenario, Gloria initially gets the second relic. Then it is better to buy the first relic for 25 shards, so the expected number of shards to spend in this scenario is $20 + 25 = 45$.

Thus, the expected number of shards to spend is $\frac{50+45}{2} = 47.5$.

H. Help BerLine

time limit per test: 5 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Very soon, the new cell phone services provider "BerLine" will begin its work in Berland!

The start of customer service is planned along the main street of the capital. There are n base stations that are already installed. They are located one after another along the main street in the order from the 1-st to the n-th from left to right.

Currently, all these base stations are turned off. They will be turned on one by one, one base station per day, according to some permutation $p = [p_1, p_2, \dots, p_n]$ ($1 \le p_i \le n$), where p_i is the index of a base station that will be turned on on the i-th day. Thus, it will take n days to turn on all base stations.

Each base station is characterized by its operating frequency f_i — an integer between 1 and 24, inclusive.

There is an important requirement for operating frequencies of base stations. Consider an arbitrary moment in time. For any phone owner, if we consider all base stations turned on in the access area of their phone, then in this set of base stations there should be at least one whose operating frequency is unique among the frequencies of these stations. Since the power of the phone and the position are not known in advance, this means that for any nonempty subsegment of turned on base stations, at least one of them has to have the operating frequency that is unique among the stations of this subsegment.

For example, let's take a look at a case of $n = 7$, all n stations are turned on, and their frequencies are equal to $f = [1, 2, 1, 3, 1, 2, 1]$. Consider any subsegment of the base stations — there is a base station with a unique frequency within this subsegment. However, if $f = [1, 2, 1, 2, 3, 2, 1]$, then there is no unique frequency on the segment $[1, 2, 1, 2]$ from the index 1 to the index 4, inclusive.

Your task is to assign a frequency from 1 to 24 to each of n base stations in such a way that the frequency requirement is met at every moment. Remember that the base stations are turned on in the order of the given permutation p.

Input

The first line of the input contains an integer t ($1 \le t \le 50$) — the number of test cases in the input. Then t test case descriptions follow.

The first line of a test case contains an integer n ($1 \le n \le 8\,500$) — the number of "BerLine" base stations.

The following line contains n distinct integers $p_1, p_2, \dots, p_n$ ($1 \le p_i \le n$) — the order in which the base stations are turned on, i. e. on the i-th day the base station with the index p_i is turned on.

It is guaranteed that a correct answer exists for all test cases in the input.

Output

Print exactly t lines, where the j-th line contains the answer for the j-th test case in the input. Print the required frequencies $f_1, f_2, \dots, f_n$ ($1 \le f_i \le 24$). If there are several possible answers, print any of them.

Example

input

```
5
3
1 3 2
3
1 2 3
1
1
10
6 10 4 2 7 9 5 8 3 1
10
2 4 6 9 1 8 10 5 3 7
```

output

```
1 3 2
10 20 10
1
2 3 4 5 3 1 3 5 4 2
1 2 3 4 5 6 7 8 9 10
```

Note

In the first test case $n = 3$ and $p = [1, 3, 2]$. The base stations can be assigned frequencies $[1, 3, 2]$.

- Day 1: only the base station 1 is turned on, its frequency is 1.
- Day 2: the base stations 1 and 3 are turned on, their frequencies are $[1, 2]$.
- Day 3: all base stations are turned on, their frequencies are $[1, 3, 2]$ (in the direction along the street).

On each day, each nonempty subsegment of turned on base stations has a base station with a unique frequency among this subsegment. It can be shown that three distinct frequencies are necessary in this test case.

I. Intriguing Selection

time limit per test: 5 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

This is an interactive problem.

You are the head coach of a chess club. The club has $2n$ players, each player has some strength which can be represented by a number, and all those numbers are distinct. The strengths of the players are not known to you.

You need to select n players who would represent your club in the upcoming championship. Naturally, you want to select n players with the highest strengths.

You can organize matches between the players to do that. In every match, you pick two players, they play some games, and you learn which one of the two has higher strength. You can wait for the outcome of a match before deciding who will participate in the next one.

However, you do not want to know exactly how those n players compare between themselves, as that would make the championship itself less intriguing. More formally, you must reach a state where there is exactly one way to choose n players with the highest strengths that is consistent with the outcomes of the matches you organized, but there must be at least two possible orderings of those n players by strength that are consistent with the outcomes of the matches you organized.

Interaction

Your program has to process multiple test cases in one run. First, it should read the integer t ($t \geq 1$) — the number of test cases. Then, it should process the test cases one by one.

In each test case, your program should start by reading the integer n ($3 \leq n \leq 100$) — the number of players to select out of $2n$ players. The sum of squares of the values of n over all test cases does not exceed $10\,000$.

Then your program can organize matches zero or more times. To organize a match, your program should print a match description formatted as ? i j — a question mark followed by two distinct numbers of players participating in the match. The players are numbered from 1 to $2n$, inclusive. Remember to flush the output after printing the match description. Then your program should read the match outcome — it will be either the greater-than character (>), if the first player in the match description has higher strength, or the less-than character (<), if the second player in the match description has higher strength.

Your program can organize at most $4n^2$ matches. After it is done organizing matches, it should print the exclamation mark (!) and continue to the next test case, or exit gracefully if this was the last test case. Remember to flush the output after printing the exclamation mark.

There must be exactly one way to choose n players with the highest strength that is consistent with the outcomes of the matches you organized, but there must be at least two possible orderings of those n players by their strength that are consistent with the outcomes of the matches you organized.

The judging program picks some distinct numbers as the strengths of all players before your program starts organizing matches and uses them to answer the requests.

Example

input

```
2
3

>

<

>

<

>

>

3

<

<

<

>

>
```

output

```


? 1 3

? 4 2

? 4 5

? 6 5

? 3 4

? 5 6

!

? 3 4

? 4 2

? 5 3

? 6 4

? 3 1

!
```

Note

In the example, the players in the first test case are sorted by strength in decreasing order. From the matches in the example output, we can deduce that players 1, 2, and 3 have the highest strength, but we do not know how the player 1 compares to the player 2.

J. Just Arrange the Icons

time limit per test: 5 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

BerPhone X is almost ready for release with n applications being preinstalled on the phone. A *category* of an application characterizes a genre or a theme of this application (like "game", "business", or "education"). The categories are given as integers between 1 and n, inclusive; the i-th application has category c_i.

You can choose m — the number of screens and s — the size of each screen. You need to fit all n icons of the applications (one icon representing one application) meeting the following requirements:

- On each screen, all the icons must belong to applications of the same category (but different screens can contain icons of applications of the same category);
- Each screen must be either completely filled with icons (the number of icons on the screen is equal to s) or almost filled with icons (the number of icons is equal to $s - 1$).

Your task is to find the minimal possible number of screens m.

Input

The first line contains an integer t ($1 \le t \le 10\,000$) — the number of test cases in the input. Then t test cases follow.

The first line of each test case contains an integer n ($1 \le n \le 2 \cdot 10^6$) — the number of the icons. The second line contains n integers $c_1, c_2, \dots, c_n$ ($1 \le c_i \le n$), where c_i is the category of the i-th application.

It is guaranteed that the sum of the values of n for all test cases in the input does not exceed $2 \cdot 10^6$.

Output

Print t integers — the answers to the given test cases in the order they follow in the input. The answer to a test case is an integer m — the minimum number of screens on which all n icons can be placed satisfying the given requirements.

Example

input

```
3
11
1 5 1 5 1 5 1 1 1 1 5
6
1 2 2 2 2 1
5
4 3 3 1 2
```

output

```
3
3
4
```

Note

In the first test case of the example, all the icons can be placed on three screens of size 4: a screen with 4 icons of the category 1, a screen with 3 icons of the category 1, and a screen with 4 icons of the category 5.

K. Key Storage

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Karl is developing a key storage service. Each user has a positive integer key.

Karl knows that storing keys in plain text is bad practice. So, instead of storing a key, he decided to store a fingerprint of a key. However, using some existing fingerprint algorithm looked too boring to him, so he invented his own one.

Karl's fingerprint is calculated by the following process: divide the given integer by 2, then divide the result by 3, then divide the result by 4, and so on, until we get a result that equals zero (we are speaking about integer division each time). The fingerprint is defined as the multiset of the remainders of these divisions.

For example, this is how Karl's fingerprint algorithm is applied to the key 11: 11 divided by 2 has remainder 1 and result 5, then 5 divided by 3 has remainder 2 and result 1, and 1 divided by 4 has remainder 1 and result 0. Thus, the key 11 produces the sequence of remainders $[1, 2, 1]$ and has the fingerprint multiset $\{1, 1, 2\}$.

Ksenia wants to prove that Karl's fingerprint algorithm is not very good. For example, she found that both keys 178800 and 123456 produce the fingerprint of $\{0, 0, 0, 0, 2, 3, 3, 4\}$. Thus, users are at risk of fingerprint collision with some commonly used and easy to guess keys like 123456.

Ksenia wants to make her words more persuasive. She wants to calculate the number of other keys that have the same fingerprint as the keys in the given list of some commonly used keys. Your task is to help her.

Input

The first line contains an integer t ($1 \le t \le 50\,000$) — the number of commonly used keys to examine. Each of the next t lines contains one integer k_i ($1 \le k_i \le 10^{18}$) — the key itself.

Output

For each of the keys print one integer — the number of other keys that have the same fingerprint.

Example

input

```
3
1
11
123456
```

output

```
0
1
127
```

Note

The other key with the same fingerprint as 11 is 15. 15 produces a sequence of remainders $[1, 1, 2]$. So both numbers have the fingerprint multiset $\{1, 1, 2\}$.

L. Lexicography

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Lucy likes letters. She studied the definition of the lexicographical order at school and plays with it.

At first, she tried to construct the lexicographically smallest word out of given letters. It was so easy! Then she tried to build multiple words and minimize one of them. This was much harder!

Formally, Lucy wants to make n words of length l each out of the given $n \cdot l$ letters, so that the k-th of them in the lexicographic order is lexicographically as small as possible.

Input

The first line contains three integers n, l, and k ($1 \le k \le n \le 1\,000$; $1 \le l \le 1\,000$) — the total number of words, the length of each word, and the index of the word Lucy wants to minimize.

The next line contains a string of $n \cdot l$ lowercase letters of the English alphabet.

Output

Output n words of l letters each, one per line, using the letters from the input. Words must be sorted in the lexicographic order, and the k-th of them must be lexicographically as small as possible. If there are multiple answers with the smallest k-th word, output any of them.

Examples

input

```
3 2 2
abcdef
```

output

```
af
bc
ed
```

input

```
2 3 1
abcabc
```

output

```
aab
bcc
```