Problem A. Task Management

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Sergey is Yandex junior software engineer for only several months, but he already completed n tasks. In order to manage tasks Yandex utilizes a special task manager system (the so called task tracker), tasks in Sergey's tracker are numbered with integers from 1 to n.

As it often happens with developers, Sergey always forgets to close the completed tasks. It is almost the time of his performance review, that's why he finally managed to close all tasks. Initially all n tasks of Sergey are open. Right now tasks in his tracker follow in the order of last change time, but Sergey wants to close them strictly in the order from 1 to n.

Sergey acts as follows. He looks through the task list from top to bottom and closes some tasks. After he reaches the end of the list, he starts once again from the beginning.

Help Sergey determine how many times he has to start looking through the list from the very beginning until the moment all n tasks become closed. The task can be closed only once.

Input

In the first line of input there follows the single integer n ($1 \le n \le 200\,000$), the number of tasks completed by Sergey.

In the second line there follows the order of tasks in the tracker.

Output

Output the only integer, the number of times he has to start looking through the list from the very beginning until he closes all the tasks in the tracker.

Examples

standard input	standard output
2 1 2	1
3 3 2 1	3
5 1 3 2 5 4	3

Note

In the first sample test Sergey closes all the tasks in the system right after the first time he looks through it.

In the second sample test during the first run through the list he closes only the task number 1, during the second run he closes only the task number 2, and during the third run he finally closes the last task number 3.

Problem B. Cross-City Communication

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Yandex has offices in 16 cities across 8 countries. It often happens that employees located in different cities have to meet and have discussion about tasks and plans on future development.

In order to make this possible each office provides several conference rooms with cross-office video-chat facilities. Conference rooms are used very intensively, so the meeting organizer has to book conference rooms in all cities involved in advance so that each meeting participant may connect with everybody at the right moment of time.

You are given the information about conference rooms availability in all offices by the day of meeting, and also m queries of concluding an hour-long meeting for employees from different cities. For each query you have choose the set of conference rooms in the given cities (in each city you must choose exactly one conference room and those conference rooms must all be free during some hour-long time interval), or determine that there is no satisfying set of conference rooms. Note that the queries are independent from each other, i.e. the answer to each query doesn't actually change the availability of conference rooms.

Input

First line of the input contains an integer c ($2 \le c \le 16$), the number of offices.

After that there follow c blocks with office description. Each block starts with the name of the city the office is located in, and then follows the number of conference rooms n_i ($1 \le n_i \le 100$). After that there follow n_i lines, each of them contains the availability schedule t_{ij} and the name of the conference room s_{ij}. The schedule t_{ij} is a string consisting of exactly 24 characters, k-th of them equals to `'X'` if the conference room is unavailable for booking during the k-th hour of a day, and to `'.'` if it is available.

In the next line there follows an integer m ($1 \le m \le 1000$), the number of queries. Each of the following m lines first contains an integer l ($2 \le l \le c$), the number of cities that should have a booked conference room, after that l city names follow. City names are space-separated.

No two conference room share the same name. No two cities share the same name. Names of conference rooms and cities are non-empty strings consisting of no more than 10 English characters.

Output

For each of the m queries output the line containing either "`Yes`" (without the quotes) and the names of conference rooms, or the "`No`" (without the quotes) if it is impossible to book the suitable set of conference rooms.

You may output conference rooms in each answer in any order. If there are several possible answers to the query, you may output any correct one.

Examples

standard input	standard output
3 Moscow 2 XXXXXXXX.X.X.X.X.XXXXX Kvartal XXXXXXXXX.X.X.X.X.XXXX Kvartet Minsk 1 XX.XXXXX........XXXXXXXX Toloka Berlin 2 XX..XXXXXXXXXXXXXXXXXX Mitte XXXXXXXXXXXXXXX.....XXX Lustgarten 4 3 Moscow Minsk Berlin 2 Moscow Minsk 2 Minsk Berlin 2 Moscow Berlin	No Yes Kvartal Toloka Yes Toloka Mitte Yes Kvartal Lustgarten
3 Moscow 1 XXXXXXXX..........XXXXX Kvartal Minsk 1 XXXXXXX..........XXXXXX Toloka Berlin 1 XXXXXX..........XXXXXXX Mitte 1 3 Moscow Minsk Berlin	Yes Kvartal Toloka Mitte

Problem C. Test Invocation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

In order to check the correctness of all the changes performed by developers, Yandex uses several automated continuous testing and build system.

Vladimir's project is being tested on n tests conveniently numbered from 1 to n. Testing is done in a very tricky manner.

System starts by invoking the root test that has the number 1. Each test including the root one first invokes all the tests it depends from, and then runs a certain check of their results, the required time to run this check for the i-th test is a_i seconds. It is known that tests form a tree-like structure, i.e. each test except the root one belongs to a dependency list of exactly one other test.

Vladimir decided to increase the stability of the whole testing process by rewriting it in such manner that each test now invokes all the tests it depends from twice, and then runs their result check once. The root test is still invoked exactly once. After his changes some tests are being run several times, and the total running time of all tests increased.

By knowing the description of all tests determine the total running time of a whole testing process on all the tests before Vladimir's changes and after.

Input

The first line of the input contains the only single integer n ($2 \le n \le 50$), the number of tests in Vladimir's system.

The second line contains n integers a_i ($1 \le a_i \le 50$), the time of checking the results in each of the tests.

The following n lines contain the description of dependencies between tests. The i-th line contains the number k_i ($0 \le k_i \le n-1$), the number of tests that the i-th test is dependent from. After that this line contains k_i integers b_{ij} ($2 \le b_{ij} \le n$), the indices of the tests that i-th test is dependent from.

It is guaranteeed that each of the tests from 2-nd to n-th appears in a dependency list of exactly one other test.

Output

In the only line output two integers: total running time of the original and new testing systems.

Examples

standard input	standard output
2 1 1 1 2 0	2 3
4 1 2 3 4 3 2 3 4 0 0 0	10 19
5 50 40 30 20 10 2 2 3 1 4 0 1 5 0	150 350

Note

In the third sample in the original testing system the checks are being run in the following order: 5, 4, 2, 3, 1. After Vladimir's change checks are being run in the following order: 5, 5, 4, 5, 5, 4, 2, 5, 5, 4, 5, 5, 4, 2, 3, 3, 1.

Problem D. Artihmetic Mean Encoding

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Ivan is a junior software engineer in the department of data transmission and storage that works on new ways of encoding. Recently he came up with an encoding algorithm that he calls an *Arithmetic Mean Encoding*.

Ivan starts with splitting the whole input stream into 50-bit integers and throwing away all the zeroes from it. After that each of t remained positive integers n_i are expressed as a mean value of k powers of two with non-negative integer exponents a_{ij}. Strictly speaking, n_i should satisfy the equality

$$n_i = \frac{2^{a_{i1}} + 2^{a_{i2}} + \ldots + 2^{a_{ik}}}{k}$$

Ivan has proven that for any n_i such an expression exists. In order to achieve the best data size statistics Ivan wants the number of elements in the expression to be as small as possible.

Help Ivan and determine the minimum number of elements a_{ij} in an expression for each of n_i and the elements themselves.

Input

The first lien of the input contains an integer t ($1 \le t \le 1000$), the number of n_i you have to process.

Each of the following t lines contains a single positive integer n_i ($1 \le n_i < 2^{50}$).

Output

For each test case output the only line. Output an integer k ($k \ge 1$) and the elements of a sequence $a_{i1}, a_{i2}, \ldots, a_{ik}$. If there are several sequences of the smallest possible length, output any of them.

Separate numbers in each line with spaces.

Example

standard input	standard output
5 1 2 10 22 32	1 0 1 1 2 2 4 3 0 6 0 1 5

Problem E. Cluster Connection

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

As an instrument of data storage and processing Yandex utilizes many special clusters. Each cluster consists of several servers connected in a single network, topology of which may be really tricky sometimes.

Administrator Konstantin gained access to the cluster consisting from n servers. By studying the cluster arrangement, he learned that the servers are connected by only $(n+1)$ bidirectional connections. Of course the cluster is connected: it is possible to transmit data from any server to any other by using a sequence of connections. Moreover, in order to make network more stable, the connectivity of a whole network is not violated when any particular connection between two servers is lost.

Konstantin thought about the following question: how many ways of connecting n servers with $(n+1)$ connections exist that satisfy the given properties? The connection may be set up between any two distinct servers, and of course there may be no more than one connection between each pair of servers. Two ways are considered different if there exists a pair of servers that is directly connected in one of them and is not connected in another.

As the answer may be pretty large, find it modulo $10^9 + 7$.

Input

The only line contains an integer n ($4 \le n \le 1000$).

Output

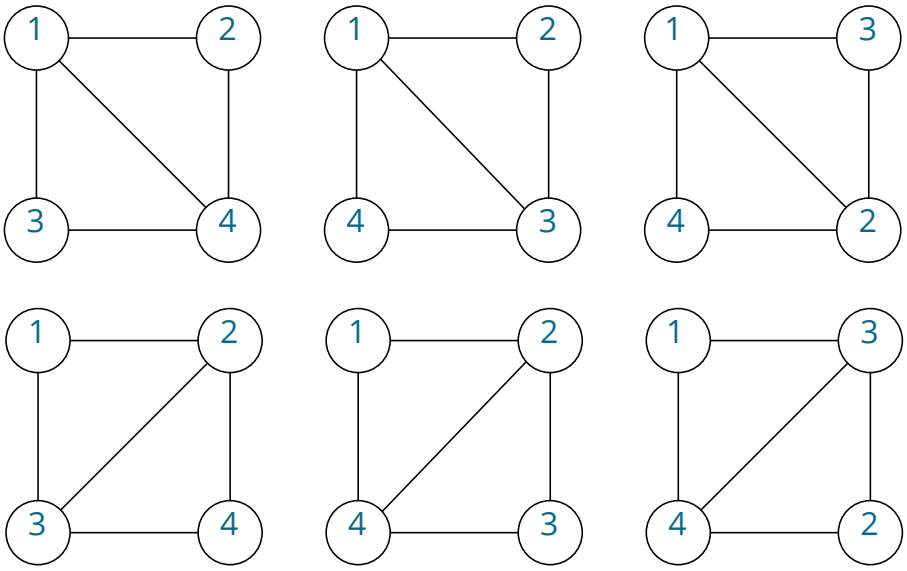
In the only line print the number of ways to connect n servers with $(n+1)$ connections such that the mentioned property of stability is satisfied modulo $10^9 + 7$.

Examples

standard input	standard output
4	6
5	85

Note

The picture below shows all the ways for the case of 4 servers.



Problem F. Repetitions

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Yandex search engine processes millions of requests per day. Thousands of servers and lots of different components work to make it operate properly. One of the most important problems that arise inside the search engine is the analysis of a repeated text patterns.

Fedor is a Software Engineer. He invented a notion of a *special pattern* in a text. Fix some text s consisting of lowercase English letters and digits. Let's call its substring z (different from s itself) special, if it has at least **three** distinct occurrences in s, among which there is a prefix of s (substring that includes the first letter of s) and a suffix of s (substring including the last letter of s). These occurrences should be distinct but may be overlapping.

Fedor is going to use the length of a longest special pattern in a text as a special factor when analyzing the content of a text. He wants to calculate the total length of a longest special pattern over all prefixes of s. If some prefix doesn't contain non-empty special patterns, consider the length of its longest special pattern be equal to zero.

Help Fedor calculate the desired value.

Input

The only line of the input contains a string s ($1 \le |s| \le 1\,000\,000$), the text that is being analyzed by Fedor. It contains only the lowercase English letters and digits.

Output

In the only line of the output print the total length of a longest special patterns in all prefixes of s.

Examples

standard input	standard output
aaaaa	6
10101101	7
abacabacabacaba	30