Multithreaded Program

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Maurice is debugging a multithreaded program on his old machine. The program has several threads operating on a set of shared variables. Each thread executes its own sequence of assignments in a predefined *program order*. Each assignment sets one of the variables to an integer value.

When the program is run, assignments from different threads can be executed in any order. The only guarantee is that each thread executes all of its assignments in the program order.

For example, let's say the program has three threads that have 2, 2, and 1 assignments in their sequences, respectively. Then one valid program execution looks as follows:

- thread 1 executes the first assignment in its sequence;
- thread 2 executes the first assignment in its sequence;
- thread 2 executes the second assignment in its sequence;
- thread 1 executes the second assignment in its sequence;
- thread 3 executes the only assignment in its sequence.

This execution can be described as $1, 2, 2, 1, 3$, where numbers specify the threads performing each assignment, in order. Note that many other valid executions are possible.

Maurice suspects that his machine is broken and can work incorrectly. He has run his program and recorded the values of all variables at the end.

Find an execution of the program that performs all assignments and leads to the recorded values of all variables, or report that the machine is indeed broken and such an execution does not exist.

Input

The first line contains a single integer t — the number of threads ($1 \le t \le 100$). The threads are numbered from 1 to t. The following lines describe t sequences of assignments, one per thread.

The first line of the i-th description contains an integer l_i — the length of the sequence of assignments of the i-th thread ($1 \le l_i \le 100$). Each of the following l_i lines contains an assignment in the form "`<variable>=<value>`". The assignments are listed in the program order. Variable names consist of up to 10 lowercase English letters, and values are positive integers not exceeding 10^9.

The first of the remaining lines contains an integer k — the number of variables ($1 \le k \le 10\,000$). Each of the following k lines contains a variable name and its recorded value, which is a positive integer not exceeding 10^9. Each variable used in the program is listed exactly once, and each listed variable is used in at least one assignment.

Output

Print "`Yes`" if an execution producing the recorded values exists, and "`No`" otherwise.

If an execution exists, print a line containing $s = l_1 + l_2 + \cdots + l_t$ integers $c_1, c_2, \ldots, c_s$, describing such an execution ($1 \le c_i \le t$). This specifies that the first assignment is performed by thread c_1, the second one is performed by thread c_2, and so on. Each thread performs its assignments in the program order. After the s-th assignment, each variable must have the recorded value. The i-th thread must appear in the description exactly l_i times.

Examples

standard input	standard output
2 2 a=1 b=2 2 b=1 a=2 2 a 1 b 1	No
3 5 start=1 counter=1111 counter=10 counter=3333 finish=1 4 start=2 counter=20 counter=10 finish=2 3 start=3 qwerty=787788 finish=3 4 counter 10 start 1 finish 1 qwerty 787788	Yes 2 3 3 2 1 1 3 1 1 2 2 1