Halting Wolf

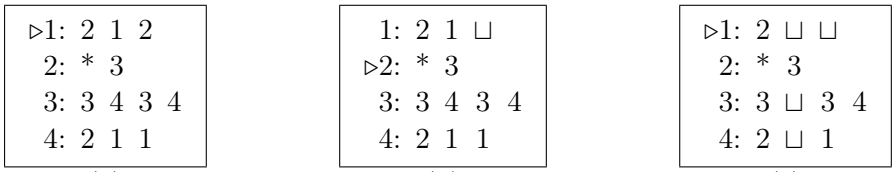
Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.25 seconds
Memory limit:	1024 megabytes

Senoof loves programming languages, and the only thing he loves more than using them is creating new ones. His latest invention is the Wolf Programming Language, a very simple language consisting of only two types of instructions. They are numbered consecutively and written one under the other to make a program. Execution starts at instruction 1 and continues until the program gets stuck.

The two types of instructions are:

- "$K\ L_1\ L_2\ \cdots\ L_K$" is a finite jump. Each value L_i is an instruction number in the program, while K indicates how many of them are specified. When a finite jump is executed, one of the values L_i is chosen, and the execution continues with instruction L_i. But that's not all! The program changes the finite jump instruction so as to consume the chosen value. If a program executes a finite jump without available values, it gets stuck and halts.
- "$*\ L$" is an infinite jump. When it's executed, the program continues with instruction L, leaving the infinite jump instruction unmodified.

I know, Senoof is crazy, but it's not that difficult. The picture below shows an example, where current instruction is indicated with a ▷ sign, and a consumed value is denoted with a ⊔ sign. The program in (a) starts execution at instruction 1, which is a finite jump. Suppose that the second value is chosen, that is, execution continues with instruction 2 and this value is consumed in instruction 1, which yields the situation shown in (b). Since instruction 2 is an infinite jump to instruction 3, execution continues with this instruction, without consuming any value from instruction 2. Now imagine that from instruction 3 execution jumps to instruction 4, then to instruction 1, and then again to instruction 1, consuming the corresponding values. The situation at this point is shown in (c). As you can see the program gets stuck and halts, because there are no available values for jumping.



After some playing around, Senoof noticed that programs written in Wolf may run forever, which does not imply that a given instruction can be executed infinitely many times. He kindly just sent us the following example of a program that may run forever, although instruction 1 can be executed at most twice.

```
1: 2 1 2
2: * 4
3: 3 4 3 4
4: * 2
```

Given a program written in Wolf, you must determine the maximum number of times that instruction 1 can be executed.

Input

The first line contains an integer N ($1 \leq N \leq 100$), the number of instructions the program has. Each of the next N lines describes an instruction. A finite jump is represented with a non-negative integer K

followed by K integers $L_1, L_2, \ldots, L_K$ ($1 \leq L_i \leq N$ for $i = 1, 2, \ldots, K$). On the other hand, an infinite jump is described with the character "*" (asterisk) followed by an integer L ($1 \leq L \leq N$). It is guaranteed that the total amount of instructions mentioned in the finite jumps is at most 10^4.

Output

Output a single line with an integer indicating the maximum number of times instruction 1 can be executed, or the character "*" (asterisk) if instruction 1 can be executed infinitely many times.

Examples

standard input	standard output
4 2 1 2 * 3 3 4 3 4 2 1 1	3
4 2 1 2 * 4 3 4 3 4 * 2	2
4 2 2 3 2 3 4 1 1 1 1	3
3 * 3 * 1 * 2	*
1 0	1