IEEE State of Mind

Problem Statement

Finite State Machines (FSMs) can be used to model systems whose output depends on the current state and its inputs. The states can represent many different systems like the on/off state of traffic lights at an intersection, the valve state of thrusters on a spacecraft, the click/key combinations entered by a user for a keyboard shortcut, or the send/receive steps in a communication protocol.

An FSM is defined by a set of states, inputs, and outputs. The states represent a known configuration of the system at a specific time. Transitioning to a new state depends on the current state and the inputs provided. This will in turn also produce an associated output for the system.

One of the uses of FSMs is to design and analyze logic circuits with memories like flip-flops. Your task is to write a program to visualize the timing diagram relating the inputs and outputs while printing the current state number to help analyze the circuit.

For the purpose of the program, you can assume that there are N states with M inputs. States are labeled with numbered subscripts in the range S_0 to S_{N-1}. The inputs are each labeled with a single uppercase character in the range 'A' to 'J', inclusive. You may assume that no letter or state number is skipped. Each state has P uniquely defined transitions to other states (the transitions are unique in the sense that no set of inputs will trigger more than 1 transition). A transition is only taken when the input conditions are satisfied. Transitions are evaluated at the beginning of each of the T discrete time steps.

Input Format

Input begins with a line containing N and M, where $1 \leq N < 100$ and $1 < M < 10$.

Then there are N lines, each describing the output the system is in a state and the transitions from this state. The first line contains a description of state S_0, the second line describes S_1, etc. These lines all have the following format:

$StateOutput\ P\ VariableExpression_1/Dest_1\ VariableExpression_2/Dest_2\ \ldots\ VariableExpression_P/Dest_P$

where

- $StateOutput$ is either a 0 or 1, representing the value that is output when the system is in this state
- P is the number of transitions from this state
- $VariableExpression_i$ is a comma-delimited list of $Variable=Value$ tokens. For a transition i to be activated, all of the input variables given in the $VariableExpression_i$ must have the values listed in this expression
- $Dest_i$, $1 <= i <= P$, is a destination state number, in the range [0..N), for transition i

Following the description of the states, is a line containing two integers, T and I, where T gives the number of timesteps ($1 <= T < 1000$), and I gives the number of the initial state ($0 <= I < N$).

The input ends with $M * T$ lines that provide the inputs for all M variables in each of the T timesteps. The first M values provide inputs for the M variables during the first timestep, the second M values provide inputs for the M variables during the second timestep, and so on.

The transitions should be interpreted as follows. Assume that we are in a state S_j. Assume further that there are four inputs, and the current input provided for the current timestep is (1,0,1,0). We map these inputs to

the variables, starting with letter 'A', so A = 1, B = 0, C = 1, and D = 0. We would then look for transitions defined on the line corresponding to state S_j. Since transitions must be unique, only one of the following *VariableExpressions* could appear in the list of expressions: `A=1`, `B=0`, `C=1`, `D=0`, `A=1,B=0`, `A=1,C=1`, `A=1,D=0`, `A=1,B=0,C=1`, etc. If such a transition is the i^{th} transition specified, the new state would be $Dest_i$. If there are no matches, the system remains in the current state.

Output Format

The output consists of a waveform. The waveform is represented by underscores, '_', for the number 0, and asterisks, '*', for the number 1.

A maximum of 16 ticks are printed together, where each time tick is 3 characters wide. The line numbers are labeled with the signal names as shown in the example.

As shown in the example below, on the first line of the grouping is the text `Tick #X` where `X` is replaced by the number of the first tick in the grouping.

Next come waveforms for each of the variables, in alphabetical order, one per line. On these lines the variable name is output, followed by five spaces, and then the waveform.

Then the waveform of the system output is displayed. This line begins with the word `OUT`, followed by 3 spaces, and then the waveform.

Finally, on the last line the numeric value of the system output is displayed. This line begins with the word `STATE`. The numeric values of system output should always be aligned with the third column of the respective tick.

If not all ticks have been displayed, then a blank line should be output, followed by the next group of ticks in the same format.

Sample Input

```
2 3
1 2 A=1,B=1,C=1/1 A=1,B=0/0
0 1 A=0/0
20 0
0 0 0
0 0 1
0 1 0
0 1 1
1 0 0
1 0 1
1 1 0
1 1 1
0 0 0
0 0 1
0 1 0
0 1 1
1 0 0
1 0 1
1 1 0
1 1 1
0 0 0
0 0 1
0 1 0
0 1 1
```

Sample Output

```
Tick #1
A   ____________**********____________**********
B   _____******_____******_____******_____******
C   ___***___***___***___***___***___***___***___
```

```
OUT   ********************__********************__
STATE  0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1

Tick #17
A    ________________
B    ________********
C    __****__****
OUT  ****************
STATE  0 0 0 0
```

Explanation

The sample input executes a parity check FSM. There are N=2 states with M=3 inputs. Since there are 3 inputs, these inputs are labeled `A`, `B`, and `C`.

State 0, S_0, outputs the value `1` (logic high). There are two transitions originating from this state. The first transition goes to S_1 when input `A`=1, `B`=1, and `C`=1. The second transition goes to S_0 when input `A`=1 and `B`=0 (that's a self-loop to the same state). `C` can take any value in the second transition.

S_1 outputs the value `0` (logic low). There is only one transition originating from this state defined in the input. This transition goes to S_0 when `A`=0.

Note that although not defined, there are implicit transitions to satisfy the other cases that are not explicitly described in the input. The implicit transitions are all self-loops to the same state.

Prior to tick t=1, we are in the initial state, S_0. The input is set to `A`=0, `B`=0, and `C`=0, therefore we take the implicit self-loop transition and stay in S_0 and output the value associated with the state, i.e. `1`.

When we start ticks t=2 through t=7, we are still in S_0. In each of these cases, we take the implicit loop to the same state and nothing changes in the output.

In tick t=8, the inputs, `A`=1, `B`=1, and `C`=1, trigger the transition to state S_1. This changes the output of our logic circuit to a `0`.

In tick t=9, the input, `A`=0, trigger the transition to state S_0. This changes the output of our logic circuit to a `1`.

The same pattern of inputs is repeated a few more times. Tick 17 begins on a new line because we limit the output to 16 ticks per line.