Problem G. Intuitionistic Logic

Input file:	`logic.in`
Output file:	`logic.out`
Time limit:	5 seconds
Memory limit:	64 megabytes

Recently Vasya became acquainted with an interesting movement in mathematics and logic called "intuitionism". The main idea of this movement consists in the rejection of the law of excluded middle (the logical law stating that any assertion is either true or false). Vasya liked this idea; he says: "Classical mathematics says that Fermat Last Theorem is either true or false; but this statement is completely useless for me until I see the proof or a contrary instance". So Vasya became a born-again intuitionist. He tries to use the intuitionistic logic in all his activities and especially in his scientific work. But this logic is much more difficult than the classical one. Vasya often tries to use logical formulae that are valid in classical logic but aren't so in the intuitionistic one.

Now he wants to write a program that will help him to check the validity of his formulae automatically. He has found a book describing how to do that but unfortunately he isn't good at programming, so you'll have to help him.

The construction starts from an arbitrary acyclic oriented graph $\mathfrak{X} = \langle X, G\rangle$. Then a partial order is constructed on X, the set of vertices of $\mathfrak{X}$: for any $x, y \in X$ we define $x \le y$ iff there exists a path (possibly of zero length) in $\mathfrak{X}$ from x to y. Next, consider the set $\mathfrak{P}$ of all subsets of X and the set $\mathcal{H} \subset \mathfrak{P}$ consisting of all $\alpha \subset X$ such that any two different x and y from α are incomparable (i.e. neither $x \le y$ nor $y \le x$). Note that $\mathcal{H}$ always contains the empty set and all one-element subsets of X. Now it is possible to define a map $\mathrm{Max} : \mathfrak{P} \to \mathcal{H} \subset \mathfrak{P}$. For any $M \subset X$ we put $\mathrm{Max}(M) = \{x \in M : \nexists y \in M : x \ne y, x \le y\}$ — the set of all maximal elements of M.

Next we define several operations on $\mathcal{H}$. For any $\alpha, \beta \in \mathcal{H}$ put $\alpha \Rightarrow \beta = \{x \in \beta : \nexists y \in \alpha : x \le y\}$, $\alpha \wedge \beta = \mathrm{Max}(\alpha \cup \beta)$, $\alpha \vee \beta = \mathrm{Max}(\{x \in X : \exists y \in \alpha, z \in \beta : x \le y, x \le z\})$, $\mathbf{0} = \mathrm{Max}(X)$, $\mathbf{1} = \varnothing$, $\neg\alpha = (\alpha \Rightarrow \mathbf{0})$, $\alpha \equiv \beta = ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$.

Now consider logical formulae consisting of the following symbols:

- Constants **1** and **0**;
- Variables — capital letters from A to Z;
- Parentheses — if E is a formula, then (E) is another;
- Negation — $\neg E$ is a formula for any formula E;
- Conjunction — $E_1 \wedge E_2 \wedge \cdots \wedge E_n$. Note that the conjunction is evaluated from left to right: $E_1 \wedge E_2 \wedge E_3 = (E_1 \wedge E_2) \wedge E_3$.
- Disjunction — $E_1 \vee E_2 \vee \cdots \vee E_n$. The same remark applies.
- Implication — $E_1 \Rightarrow E_2$. Unlike the previous two operations it is evaluated from right to left: $E_1 \Rightarrow E_2 \Rightarrow E_3$ means $E_1 \Rightarrow (E_2 \Rightarrow E_3)$.
- Equivalence — $E_1 \equiv E_2 \equiv \cdots \equiv E_n$. This expression is equal to $(E_1 \equiv E_2) \wedge (E_2 \equiv E_3) \wedge \cdots \wedge (E_{n-1} \equiv E_n)$.

The operations are listed from the highest priority to the lowest.

A formula E is called *valid* (in the model defined by $\mathfrak{X}$) if after substitution of arbitrary elements of $\mathcal{H}$ for the variables involved in E it evaluates to **1**; otherwise it is called *invalid*.

Your task is to determine for a given graph $\mathfrak{X}$ which formulae from a given set are valid and which invalid.

Input

The first line contains two integers N and M separated by a single space — the number of vertices ($1 \le N \le 100$) and edges ($0 \le M \le 5000$) of $\mathfrak{X}$. The next M lines contain two integers s_i and t_i each — the beginning and the end of i-th edge respectively. The next line contains K ($1 \le K \le 20$) — the number of formulae to be processed. The following K lines contain one formula each. A formula is represented by a string consisting of tokens `0`, `1`, `A`, ..., `Z`, `(`, `)`, `~`, `&`, `|`, `=>`, `=`. The last five tokens stand for $\neg$, $\wedge$, $\vee$, $\Rightarrow$ and $\equiv$ respectively. Tokens can be separated by an arbitrary number of spaces. No line will be longer than 254 characters. All formulae in the file will be syntactically correct. Also you may assume that the number $H = |\mathcal{H}|$ of elements of $\mathcal{H}$ doesn't exceed 100 and that $\sum_{1 \le j \le K} H^{v_j} \le 10^6$ where v_j is the number of different variables used in j-th formula.

Output

The output file must contain K lines — one line for each formula. Write to the j-th line of output either `valid` or `invalid`.

Example

logic.in	logic.out
1 0	invalid
6	valid
1=0	valid
X|~X	valid
A=>B=>C = (A&B)=>C	valid
~~X => X	valid
X => ~~X	
(X => Y) = (Y | ~X)	
6 6	invalid
1 2	invalid
2 3	valid
2 4	invalid
3 5	valid
4 5	invalid
5 6	valid
11	valid
1=0	invalid
X|~X	valid
A=>B=>C = (A&B)=>C	valid
~~X => X	
X => ~~X	
(X => Y) = (Y | ~X)	
A&(B|C) = A&B|A&C	
(X=>A)&(Y=>A) => X|Y=>A	
X = ~~X	
~X=~~~X	
~X = (X => 0)	