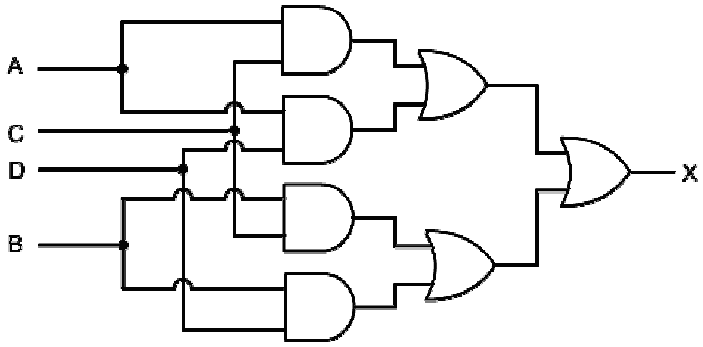
Problem E:

Evaluating Logic Expressions



Logic expressions occur frequently in computer programs. The elements of logic expressions are:

- variables, which may have the values true or false
- unary and binary logic operators
- parentheses which may affect the order in which the operations are carried out

Unary operators operate on one variable, whereas binary operators operate on two variables. A common unary logic operator is NOT; some common binary logic operators are AND, OR, XOR, NAND and NOR.
A logic operator can be defined by a 'truth table' (one-dimensional for unary operators, two-dimensional for binary operators). For examples, see the diagram.

Truth table for NOT

Operands →	false	true
Results →	true	false

Truth table for NAND
(operands across top and on left margin)

	false	true
false	true	true
true	true	false

Note that false comes before true in the header across the top (for unary or the right operand of binary operators), as well as on the left margin (for the left operand of binary operators).
Two examples of logic expressions are:

1. (x AND (NOT(y NAND z)))
2. (x OR ((NOT y) XOR z))

For the purposes of this problem, the precise structure of a logic expression is defined by the grammar:

<expression> = <variable> | (<expression> <operator> <expression>) | (<operator> <expression>)
<variable> = <lowercase_letter>
<operator> = <uppercase_letter> | <operator> <uppercase_letter>

(Here the vertical bar '|' is pronounced 'or' and is used to define the grammar; it does not actually show up in the expression.) <lowercase_letter> and <uppercase_letter> have their usual meanings.

In some cases, it is possible to evaluate a logic expression even when not all of the variables have been assigned values. Consider Example (1) above, and suppose that y = false, but the values of x and z are not known. It can be seen that the given expression evaluates to false regardless of the values of the unassigned variables. On the other hand, suppose that, in the same Example (1), x = true, y = true, and z is unknown. it is not possible to determine the value of the expression without knowing the value of z.

Input Format

The input will contain data for one or more test cases. For each test case, the first line of input will contain two non-negative integers (not exceeding 100), the number of unary operators and the number of binary operators to be considered for that case. The first line will be followed by several lines that will name each operator and define it in the form of a truth table, as described in the next two paragraphs. The names of the operators do not exceed 20 characters and are unique for each operator.

First, each unary operator will be defined in two lines of input: the first of these two lines will contain the name of the operator; the second line will contain two

true / false entries that define the table for the unary operator, without the implied column headers across the top.
After the unary operators have been defined, each binary operator will be defined in three lines of input: the first of these three lines will contain the name of the operator; the second and third lines will each contain two true / false entries. These two lines define the table for the binary operator, without the implied column headers across the top and row headers on the left margin.
The operator tables will be followed by a line containing a valid logic expression satisfying the above grammar. Variables will be separated from adjacent logic operators by one or more blank spaces. Parentheses may or may not be separated from adjacent elements of the expression by blank spaces. You may assume that no variable will occur more than once in an expression. The expression will consist of at least 1 but not more than 500 characters.
The expression will be followed by zero or more lines that comprise a table of values. Each of these lines will have one of the two forms:

<variable> true
<variable> false

No variable will appear more than once in the table.
The end of each test case will be marked by a line containing a single asterisk.
End of input will be marked by a line containing two negative integers.

Output Format

For each test case there will be one line of output. The case number will be printed in the format of the sample output.
It will be followed by one of the words: true, false, or unknown; whichever is appropriate for the given expression, as explained above.
The output format is illustrated in the sample output.

Sample Input

1 2
NOT
true false
AND
false false
false true
TWEEK
true false
true false
(x AND (NOT(y TWEEK z)))
x true
y true
*
1 1
MOCK
true true
NAND
true true
true false
(x NAND (MOCK (y NAND z)))
x false
y false
*
0 2
XOR
false true
true false
FAKE
true true
false false
((p XOR q) FAKE r)
p true
q false
*
-1 -1

Sample Output

```
Case 1: unknown
Case 2: true
Case 3: false
```

Howard Cheng

ACPC 2012