Problem 15: The Pudding[1]

Source filename:	`pudding.(cpp|java)`
Input filename:	`pudding.in`
Output filenames:	`pudding.out`

Formal Logic

In formal logic, each step in a proof must adhere to a very precise rule of inference. A rule of inference is made up of a series of conditions which must be satisfied in order to use the rule, followed by the conclusions that result. For example, one well known rule of inference is **modus ponens**, which is written like this:

$$\frac{\mathbf{F} \supset \mathbf{G} \qquad \mathbf{F}}{\mathbf{G}}$$

In this notation the conditions of the rule of inference are written on top of the line, and the result is written below the line. This rule has two conditions. The first condition, written as "**F ⊃ G**", is read, "**F** implies **G**," meaning that if **F** is true then **G** is also true. The second condition, written as "**F**", means that **F** is true. The result of this rule is that **G** is now accepted as true.

Note that **F** and **G** are variables, they can stand for anything. So, for example, if we knew that:

- It is raining ⊃ the sidewalk is wet
- It is raining

Then we can meet the conditions of the rule by letting **F** stand for "it is raining" and **G** stand for "the sidewalk is wet". This means we can safely conclude that the result (**G** by itself) is true:

- The sidewalk is wet

For this problem you will be given a rule of inference (made up of 1 or more conditions plus 1 result), followed by several statements which are assumed to be true. Your goal is to write a program that reads a data set from a file and prints to an output file all of the statements that can be proved by *just one* application of the rule of inference.

The Input File (pudding.in)

All statements for this problem will be written as **formulas**, where a formula is defined as a whitespace-delimited list consisting of symbols and variables written within a pair of parentheses. A **symbol** is a sequence of alphabetic characters beginning with a *lowercase* letter, and a **variable** is a sequence of alphabetic character beginning with an *uppercase* letter. A particular variable or symbol may occur more than once within the same formula. The number of characters in a symbol or variable will never exceed 20. Furthermore, the number of symbols plus the number of variables in a formula will never exceed 20. Thus, counting spaces and parentheses, a formula will never exceed 423 characters.

For example, here is a formula consisting of three symbols:
```
( raining implies wetSidewalk )
```

Here is a formula consisting of a variable, a symbol, and another variable.
```
( F implies G )
```

The input file contains 1 or more data sets. The format of a data set is as follows:

- The first *n* lines *(1 ≤ n ≤ 10)* contain formula*(s)* (one per line) that represent the ***conditions*** of the rule of inference. The total number of variables contained in these *n* lines will not exceed 4.

[1] This problem appeared at the CCSC:MidSouth Student Programming Contest April 4, 2008.

- The line following the n^th formula contains a single hyphen "-" character (ASCII 45), indicating the end of the condition formula(s).
- The next line contains a single formula representing the ***result*** of the rule of inference. Note that this formula will not contain any variables that are not found in at least one of the conditions.
- The next *m* lines ($1 \leq m \leq 10$) contain formula(s) (one per line) that represent the statements which are given to be true. We'll call these the "***givens***". Note that these formulas will not contain *any* variables at all.
- The word "end" on a line by itself indicates the end of the data set. This line is followed by a blank line.
- The line after the blank line is either the start of the next data set or another line that contains the word "end". If the latter, the end of the file has been reached.

When reading the input file, note the following things:

- A formula will *always* be contained within parentheses
- There will *not* be any empty formulas; a formula will always have at least one symbol or variable.
- *All* tokens in the input file will be separated from other tokens by at least one whitespace character. That includes the parentheses, symbols, variables, the hyphen character, and the word "end".

Example Input:

```
( F )
( F implies G )
-
( G )
( rain implies wetSidewalk )
( snow implies noSchool )
( noSchool implies happyStudents )
( snow )
end

( F implies G )
( not G )
-
( not F )
( rain implies wetSidewalk )
( rain )
( not snow )
end

( A or B )
( not A )
-
( B )
( right or wrong )
( black or white )
( not white )
( not right )
( right or left )
( not rain )
end

( A )
( B )
-
( A to B )
( oranges )
( apples )
end

end
```

The Output File (pudding.out)

For each data set, you are to calculate the formulas which can be proved by applying the rule of inference to the givens *one time*. In order to prove a formula we need to find a mapping from variables to symbols that, when applied to each variable in the condition, makes it identical to a formula in the list of givens. For example, look at data set 1 above and consider the following mapping:

```
F → snow
G → noSchool
```

Let's apply this mapping to our two condition formulas. In this case the first condition becomes `( snow )` and the second condition becomes `( snow implies noSchool )`, both of which can be found in our list of givens. To find the formula that is proven, we simply apply this mapping to the variables in the result to get `noSchool`.

Note that any given may be used to match any condition, and the same given can be used to match multiple conditions. Formulas which have been proved should *not* be used as a given. For example, in data set 1, after we prove `noSchool` it would theoretically be possible to use this result with `( noSchool imples happyStudents )` to prove `happyStudents`. However, you should *not* do that for this problem; use only the givens listed in the input file.

When you print a formula, simply print the symbols all on one line, separated by a single space. Do not print parentheses; do not print anything before the first symbol or after the last symbol on the line. The output for a data set should be as follows:

- The line "`Proof #N`" where `N` is the number of the data set (with the first data set being 1).
- The formulas that can be proved from that data set, one per line. If there is more than one formula to be printed, list them in sorted order (according to standard ASCII collating sequence.) If no formulas can be proved from that data set, then simply print the line "`No matches`".
- If there is another data set, print a blank line to separate this data set's output from the next.

When generating the output, note the following things:

- The output formulas do not have any parentheses
- Each formula should be on its own line
- The symbols in a formula should be separated by a single space
- There should not be anything before the first symbol or after the last symbol
- If there is more than one formula, they should appear in sorted order (according to ASCII collating sequence).
- *Preserve* duplicates; that is, if the same formula can be proved multiple ways it should appear multiple times in the output.
- There should 1 blank line separating each data set's output; however, there should *not* be a blank line before the first data set nor after the last.

Example Output:

```
Proof #1
noSchool

Proof #2
No matches

Proof #3
left
wrong

Proof #4
apples to apples
apples to oranges
oranges to apples
oranges to oranges
```