ACM International Collegiate Programming Contest

Western Europe Regional Contest

’s-Hertogenbosch, 13–14 November 1999



Problem F

Basically Boolean

Input: f.in

Boolean expressions of more than two or three variables can become very complex. Sometimes they become so complex, it is no longer simple to see how the outcome of the expression depends on the value of a single variable. It may even be so that the outcome of an expression no longer depends on the value of one of the variables in it, i.e. that the outcome is the same whether the variable is true or false. If an expression *E* no longer depends on a variable *X*, we say that *X* is *not important* for the expression, and conversely, if a variable *X* is important for an expression E, then the outcome of *E* will depend on the value of *X*. If none of the variables in an expression are important, then the expression is either a tautology (always true), or a contradiction (always false).

You are asked to write a program that will, given an expression *E*, determine all the important variables in the expression.

Input

The first line of the input contains the number of runs *R* (1 = *R* = 1,000). Then *R* lines appear, with each line containing one formula. Each formula consists of one or more of the following elements:

- A variable, denoted by a capital letter **A** through **J** (each variable may occur multiple times, or may not occur at all)
- The unary operator **not**
- The binary operators **or**, **and**, or => *(implies)*
- Brackets: **(** and **)**, always matching

A formal description of an input line is given by the following grammar (where ‘space’ denotes exactly one space, and FORMULA is the start symbol, as shown by the underscoring):

```
FORMULA  =  VARIABLE                                          |
            UNARY_OP space FORMULA                            |
            ( FORMULA space BINARY_OP space  FORMULA )        |
VARIABLE =  A|B|C|D|E|F|G|H|I|J
UNARY_OP =  not
BINARY_OP = or|and|=>
```

Note: the **not** operator binds stronger than the binary operators, and between the binary operators no precedence needs to be defined as every formula is fully bracketed. The Boolean operators are to be interpreted as usual:

x **or** y is true, if and only if x is true, or y is true, or both are true;
x **and** y is true, if and only if x is true and y is true;
x => y is true, if and only if x is false or y is true.

Output

For each test case, you must output a line with the variables from the input expression that are important for that expression, preceded by the string ‘`Important variables: `’. Multiple variables are to be separated by single spaces. There must also be exactly one space between the ‘:’ and the first variable. If no variables are important for the evaluation of the formula, you must print the string ‘`No variables are important.`’

Sample Input

```
4
H
(not A and not not A)
(((B and C) or (D and B)) => (A or B))
((not (not E or (B or C)) or G) and not (((not E or B) or C) and G))
```

Output for the Sample Input

```
Important variables: H
No variables are important.
No variables are important.
Important variables: B C E
```