LEXSIM - Sintatic and Lexical Avaliator

By Neilor Tonin, URI Brazil

Timelimit: 1

One of the most interesting use of stack is in evaluation of a mathematical expression. We can, through the stack, do the lexical analysis of the expression (indicating whether a valid expression has an invalid operand -a symbol which is not present in the table of operators or a symbol which is not present in the table of operands) and also the parsing. The parsing may indicate that it is missing one or more brackets, leaving one or more brackets, leaving the operator, two successive operands, etc.. The task here is to determine whether an expression is correct or is not correct.

Input

In the input, are valid:
a) Operands: all letters uppercase and lowercase ('a'..'z', 'A'..'Z') and numbers (0...9).
b) Parenthesis.
c) Operators: all following operators in the priority table shown below are accepted:

Operator	Priority
^	6
*,/	5
+,-	4
>,<,=,#,	3
AND (.)	2
OR (|)	1

To make your job a little easier, you must use the point like AND (.) e the pipe like OR (|).

Not will allowed expressions with unary operators. Example: 4 * -2

The end of input is determined by the end of file EOF().

Output

As output, for each input expression should be generated a line indicating the result of processing. If the expression is correct, it should be transformed to the form posfix. If not possible, messages should be printed indicating Lexical Error or Syntax Error, in this order.

Sample Input	Sample Output
((A+ (A+B)*c (A+B)*% (a+b*c)/2*e+a	Syntax Error! Syntax Error! AB+c* Lexical Error! abc*+2/e*a+

(a+b*c)/2*(e+a)	abc*+2/ea+*
(a+b*c)/2*(e+a	Syntax Error!
(ab+*c)/2*(e+a)	Syntax Error!
(a+b*cc)/2*(e+a	Syntax Error!
(“a+b*cc)/2*(e+a	Lexical Error!
a+b-c	ab+c-
a-b*c/d+e	abc*d/-e+