Problem A
Cerl

Input file: *testdata.in*
Time limit: 1 second

Problem Description

Cerl is an "old-fashion" programing language, and it appears in this modem era. Dr. Cerl want his students know how to design it. Help the students of Dr. Cerl to write a program which reads lines of text and recognizes the Cerl tokens in them.

Technical Specification

1. The only data type is integer.
2. All identifiers are implicitly declared and are not longer than 32 characters.
3. Identifiers are composed of letters, digits and underscores.
4. At least one character of the identifiers is not a digit.
5. Literals are strings of at most 8 digits.
6. Comments begin with -- and end at the end of the line in which they start.
7. Statement types are
   (a) Assignment:
      i. <identifier> := <expression>

- where expressions are constructed from identifiers, literals, operators +, -, and parentheses as follows:
  - all identifiers and literals are expressions,
  - if a and b are expressions then a + b, ab, +a, a, (a) are expressions.

i. Input/Output:
- read (List of identifiers)
- write (List of expressions)
- (Items in the list are separated by comma)

8. begin, end, read, and write are reserved words.
9. Each statement is terminated by a semicolon.
10. Cerl is case-sensitive, for example BegIN is not the same keyword as beGin.
11. Cerl tokens are defined to be the identifiers, the literals, or the following symbols:
    - `+`
    - `-`
    - `(`
    - `)`
    - `:=`
    - `;`
    - `,`

- Notes: the assign operator is to be considered one Cerl token; spaces, tabs, and end-of-lines are allowed between the tokens; no part of any comment is a token; successive tokens that are either identifiers, literals, or reserved words must be separated by a space, a tab, or end-of-line; no token is allowed to contain a space, a tab, or end-of-line.

Input Format

- The input file consists of several blocks of lines.
- Each block contains lines of text and is terminated by one empty line.

Output Format

- The output file consists of blocks corresponding to the blocks in the input file.
- In the lines of each block there are successively stored the Cerl tokens recognized by the program (just one token on each line).
- Each token must be written on the output line in exactly the same form as it appears in the input text.
- If the program encounters a string that is neither a Cerl token, nor comment, nor space, tab, end-of-line, it is to write the string `TOKEN_ERROR` on a new line and continues by processing the next block in the input file.
- The program writes one empty line after each block of the output file.

Sample Input

```
A1:= A+(-B)

A123_A123 )
01.2 A B
C

:= A beGIn

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

Sample Output

```
A1
:=
A
+
(
-
B
)

A123_A123
)
01
TOKEN_ERROR

:=
A
beGIn

TOKEN_ERROR
```