# 309 FORCAL

FORCAL is the programming language well known to programmers who are interested in compiler construction and especially to students attending Dr. C. Ompiler class. The FORCAL syntax is defined as follows:

- The only data type is integer.
- All identifiers are implicitly declared and are not longer than 32 characters. Identifiers are composed of letters, digits and underscores. At least one character of the identifier is not a digit.
- Literals are strings of at most 8 digits.
- Comments begin with – and end at the end of the line in which they start.
- Statement types are

  Assignment:
  ⟨identifier⟩ := ⟨expression⟩
  where expressions are constructed from identifiers, literals, operators +, -, and parentheses as follows:
  1) all identifiers and literals are expressions,
  2) if $a, b$ are expressions then $a + b$, $a - b$, $+a$, $-a$, $(a)$ are expressions.

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

- **begin**, **end**, **read**, and **write** are reserved words.
- Each statement is terminated by a semicolon.
- FORCAL is not case–sensitive, for example **BegIN** is the same keyword as **beGin**.

FORCAL tokens are defined to be: the identifiers or the literals or the symbols `+ - ( ) := ; ,` or the reserved words.

**NOTES:**

- the assign operator is to be considered one FORCAL token,
- spaces, tabs, 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 or tab or end-of-line,
- no token is allowed to contain a space or a tab or end-of-line.

Help the students of Dr. C. Ompiler to write a program which reads lines of text an recognizes the FORCAL tokens in them.

## Input

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

## Output

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 FORCAL 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 FORCAL 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
```