OPHP (3) P21501_es

Seguimos con el diseño del OPHP (por ahora, no será necesario que hayas resuelto los problemas anteriores '`OPHP1`' y '`OPHP2`'). En este problema te pediremos que hagas un *parser* de expresiones matemáticas de OPHP. En concreto, tu programa deberá leer líneas de texto e identificar los siguientes *tokens*:

- Variables ('`VAR`'): un símbolo '`$`' seguido de una secuencia no vacía de letras mayúsculas, minúsculas, dígitos o el carácter de subrayado.
- Números ('`NUM`'): una secuencia no vacía de dígitos.
- Operaciones aditivas ('`OP+`'): los símbolos '+', '−'.
- Operaciones multiplicativas ('`OP*`'): los símbolos '`*`', '`/`', '`%`'.
- Otros símbolos: paréntesis abierto '`(`', cerrado '`)`' y el símbolo igual '`=`'.

Tu *parser* tiene que ir leyendo la entrada, letra a letra, y listar los *tokens* que vaya encontrando, ignorando los espacios. Si tu programa encuentra una combinación de símbolos que no puede ser parseada correctamente, deberá escribir el *token* '`ERROR`', y dejar de parsear la línea.

Entrada

Una secuencia de líneas a parsear.

Salida

Para cada línea de la entrada, escribe en una línea y separados por espacios los *tokens* que vayan apareciendo. Ignora lo que queda de línea en caso de encontrar un '`ERROR`'.

Ejemplo de entrada 1

```
$a $b $c ($hola 31$1hola 3$1 hola)
$a$b$c  (  ( (()()
123$a123$b $124 $_   12$q 12$q12
+- $a ++ 123 ( -*/%) ()))

   $123 123$123$$abc
$a=$b*(1+2*(1+$1))
( 123*(1/3%$3)-a+(123+123)
  _$_
       $123+123$123($1*$2)
```

Ejemplo de salida 1

```
VAR VAR VAR ( VAR NUM VAR NUM VAR ERROR
VAR VAR VAR ( ( ( ) ( )
NUM VAR VAR VAR VAR NUM VAR NUM VAR
OP+ OP+ VAR OP+ OP+ NUM ( OP+ OP* OP* OP* ) ( ) ) )

VAR NUM VAR ERROR
VAR = VAR OP* ( NUM OP+ NUM OP* ( NUM OP+ VAR ) )
( NUM OP* ( NUM OP* NUM OP* VAR ) OP+ ERROR
ERROR
VAR OP+ NUM VAR ( VAR OP* VAR )
```

Información del problema

Autoría: Omer Giménez

Generación: 2026-01-25T10:12:11.242Z