While Language

Here you have to design an interpreter for a subset of the *While* language. It is a simple imperative language which only supports integer literals.

We will use similar grammar which its authors[1,2,3] have used. Below is the description of grammar that we will use.

- x, y ∈ **Var** (variables)
- n ∈ **Num** (numerals/integers)
- op_a ∈ $\mathbf{Op_a}$ (arithmetic operators)
  op_a ::= `+` | `-` | `*` | `/`
- op_b ∈ $\mathbf{Op_b}$ (boolean operators)
  op_b ::= `and` | `or`
- op_r ∈ $\mathbf{Op_r}$ (relational operators)
  op_r ::= `>` | `<`
- a ∈ **AExp** (arithmetic expressions)
  a ::= x | n | a_1 op_a a_2 | (a)
- b ∈ **BExp** (boolean expressions)
  b ::= **true** | **false** | b_1 op_b b_2 | a_1 op_r a_2 | (b)
- S ∈ **Stmt** (statements)
  S ::= x := a | S_1 **;** S_2 | **if** b **then** **{** S_1 **}** **else** **{** S_2 **}** | **while** b **do** **{** S **}**

Here all operators are left associative. Their *precedence order* is as follows.

1. *Arithmetic Operators:* (`*`, `/`) > (`+`, `-`) > (`>`, `<`)
2. *Boolean Operators:* `and` > `or`

You can safely assume that all variables have integer type and are initialized properly. All variables name will consist of only lowercase letter ('a'-'z') and it's length will not exceed 10.

Note that "`;`" is more like of a sequencing operator. It is used to concatenate two statements. That's why there will be no "`;`" at the end of block of statements.

All divisions are integers divisions, that is, `a/b = floor(a/b)`. Intermediate values of any variable will always be in range [0, $2*10^{18}$].

All test cases are *valid* programs. All of them will execute no more than 10^6 operations. All operators and operand will be separated by at least one white space.

Input
Input will be the multiline *While* program. You have to read it to the end of file.

Output

At the end of program, you have to print each variable's name and its value, in different lines, sorted by the lexicographical order of name.

Sample Input #00

```
fact := 1 ;
val := 10000 ;
cur := val ;
mod := 1000000007 ;

while ( cur > 1 )
  do
   {
      fact := fact * cur ;
      fact := fact - fact / mod * mod ;
      cur := cur - 1
   } ;

cur := 0
```

Sample Output #00

```
cur 0
fact 531950728
mod 1000000007
val 10000
```

Sample Input #01

```
a := 10 ;
b := 100 ;

if ( a < b ) then
    {
        min := a ;
        max := b
    }
else {
    min := b ;
    max := a
    }
```

Sample Output #01

```
a 10
b 100
max 100
min 10
```

Explanation

Sample Case #00: This programs calculate the factorial of a number. Here it calculate the value of `10000! % (10^9+7)` using *while* statement. Using the property `a % b == a - (a/b)*b` we calcuated the modulo of solution.

Sample Case #01: This program finds the maximum and minimum of `a` and `b` using *if else* statement.