Intuitive language

Sometimes it is hard to read code written in the language you are not familiar with, not unless its written in Intuitive Language. Here's the sample of a program.

```
A is 15.
Sum is function of 2: 1, 1, 0.
Inc is function of 1: 1, 1.
I is 1.
F1 is 1.

do {10} assign I*F1 to F1 AND Inc[I] to I!
what is I AND F1?

what is Sum[1]?
```

Pretty straightforward, when compared with some popular languages, isn't it? Lets break down the rules here.

Language allows:

```
* To declare function.
* Assign value to variable.
* Make a loop with fixed number of repetition.
* Ask about function's value.
```

How to declare a function?

Function is an expression of `n` parameters.

`Function`$_1$ `is function of n:` k_1, k_2, `...`, k_n, k_0.

represents a function called `Function`$_1$, when called with `n` parameters it will yield following value

k_1`*%1 +` k_2`*%2 + ... +` k_n`*%n +` k_0

where `%i` is the i^{th} parameter, and k_0, k_1, `...`, k_n are the coefficients.

Formally, a function with `n` parameters is defined as

`%Variable% is function of %N%: %Expression`$_1$`%, %Expression`$_2$`%, ..., %Expression`$_N$`%, %Expression`$_0$`%.`

where k_0 `= Expression`$_0$, k_1 `= Expression`$_1$, `...`, k_n `= Expression`$_n$.

Simpler, optional, way to declare a function with no $(n = 0)$ parameter is:

`%Variable% is %Expression%.`

Note:
1. Declaration ends with a dot, ('`.`') .
2. You can declare a function, with one or more parameters, only one time.
3. It's guaranteed that function declaration will not any contain function calls, i.e., there will be no coefficient k_i which contains a function call.

4. A function with 0 parameter will be considered as a variable. It can be reassigned multiple times.

Examples
1.

```
A is function of 2: 3, -15, 1.
```

That means `A = 3*%1 - 15*%2 + 1`.
2.

```
B is function of 0: 14.
```

That means `B = 14`.
3.

```
C is 7.
```

That means `C = 7`.

How to assign value to variable?

`Assign %Expression₁% to %Variable₁% [ AND %Expression₂% to %Variable₂% ... ]!`

This means that the value of $\texttt{Variable}_i$ will be $\texttt{Expression}_i$.

Note:
1. Multiple assignments go left-to-right.
2. Assignment ends with an exclamation sign, '`!`'.
3. You can assign values to a variable multiple times.

Examples
1.

```
assign (3+4*2) to V!
```

`V` will be equal to 11.
2.

```
assign 3 to P AND (4+5) to Q AND (Q-15) to R!
```

This expression assigns 3 to `P`, 9 to `Q` and -6 to `R`.

How to make a loop?

`do {Expression} %Assign values(s) to variable(s)%!`

Assignment inside the loop will be repeated number of times, indicated in `{}`. This number is guaranteed to be positive integer.

Examples

```
I is 1.
F is 1.
do {3} assign I*F to F AND I+1 to I!
```

Initially it will be `I = F = 1`. It will loop 3 times. At first loop, `F = 1, I = 2`, after second loop `F = 2, I = 3` and after third/final loop it will be `F = 6, I = 4`.

How to ask about variables and function's value?

`what is %Function₁ call% [AND %Variable₁% [ AND %Function₂ call% ... ] ]?`

- A function's value or variable's value can be asked anywhere, once it has been initialised.
- *Function call:* `%Name%[Expression₁][Expression₂]...[Expression_M]`.
    - `Expression_i` will be the i_{th} parameter, `%i`, in function declaration.
- Results
    - If all `n` parameters are provided: the result of function call will be a value.
    - Otherwise: it will result into another function which takes remaining parameters. You have print coefficients, comma and space separated, as output in a line.
- Question ends with '`?`'.

Examples

```
A is 15/4.
Sum is function of 2: 3, 2, 4.

what is Sum[10] AND A?
what is Sum[20][10]?
```

will print the following output

```
2, 34
15/4
84
```

Let's represent *Sum* function as

$$\lambda(a, b) \rightarrow 3 * a + 2 * b + 4$$

So, *Sum[10]* ($a = 10$) will result into another function

$$\lambda(b) \rightarrow 3 * 10 + 2 * b + 4$$
$$\Rightarrow \lambda(b) \rightarrow 2 * b + 34.$$

So new coefficients will be `2, 34`.

Calling *Sum[20][10]* will result into `3*20 + 2*10 + 4 = 84`

Notes:

1. All numbers are rational integers, represented like a / b (or just a if b=1). Here b is a positive integer, a is any integer. Greatest common divisor for a and b should be 1.
2. - *Number:* n where n is any integer number.

- *PNumber: n* where $n > 0$.
- *Value: Number* | *Number / PNumber* | *FValue*.
- *FValue:* Calling function of *n* with all *n* parameters.
- OP_a: + | -.
- OP_b: * | /.
- *Expression: Term [*OP_a $Term_1$ *...]*.
- *Term: Fact [*OP_b $Fact_1$ *...]*.
- *Fact: Value* | *(Expression)*
- *Letter:* 'A'..'Z' | 'a'..'z'.
- $OP_b > OP_a$

3. Variable consists of 1 or more letters followed by 0 or more digits; Key words (`is`, `of`, `assign`, `what`, `function`, `do`, `and`, `to`) are not allowed as variable names.
4. Language is case insensitive. Variable name, function name and keywords are all case insensitive.
5. The coefficients of functions are guaranteed to be expressions without variables.
6. It is guaranteed that function call will occur only after it's definition.
7. Whitespace between tokens should be ignored.
8. Operators
   - *Operator precedence:* `*` = `/` > `+` = `-`
   - *Associativity:* All operators follows left associativity.

Constraints

- It's guaranteed that function declaration will not contain function calls.
- All given numbers will be between -100 and 100.
- Resulting numerator / denominator will not exceed 10^9 by absolute value.
- Functions will have no more than 5 parameters.
- Function name will contain no more than 20 characters.
- Each loop will have no more than 1000 iterations.

Input
You are given compilable working program written in IL. You have to read it to the end of file.
It will contain no more than 100 lines.
Each line will contain no more than 200 characters and will represent one statement.

Output Format
For each function call/variable in *what is* question output it's value, each per line.
If function is provided with all parameters that it was declared with - the result will be value, otherwise - function with remaining parameters.

Output Value Format
Print *a* / *b* (without spaces) if $b > 1$ or just *a* if $b = 1$.

Output Function Format

Resulting function of n parameters should be printed in the following format:

$k_1, k_2, \dots, k_n, k_0$

where k_i is the coefficient.

Sample Input #00

```
A is 15.
Sum is function of 2: 1, 1, 0.
Inc is function of 1: 1, 1.
I is 1.
F1 is 1.

do {10} assign I*F1 to F1 AND Inc[I] to I!
what is I AND F1?

what is Sum[1]?
```

Sample Output #00

```
11
3628800
1, 1
```

Sample Input #01

```
A is 2/6.
b Is function of 2: -3, 7/4, 6/1.
I is 1.

what is b[0] and A and b[-1/3]?
do {b[-1][4]-A*30} assign I+1 to I and I-1 to I and I+1 to I!

what is I?
```

Sample Output #01

```
7/4, 6
1/3
7/4, 7
7
```

Explanation

Test case #00: Inside the loop we calculate 10!, so I will be 11 and 10! is 3628800. Sum function is x+y, so Sum[1] is y+1.

Test case #01:

```
A = 1/3
b = -3*x+7/4*y+6.
b[0] = 7/4*y+6.
b[-1/3] = -3*(-1/3)+7/4*y+6 = 7/4*y+7
b[-1][4] = -3*(-1)+(7/4)*4+6 = 16.
b[-1][4]-A*30 = 16-10 = 6
```

We repeat 6 times increasing of I, so I will be 7.

Tested by: Stimim Chen