Interpreter P33564_en

Olimpiada Informática Española — Final 2007 (2007)

Your task is to write a program that simulates the execution of a program written in a certain assembly language (called affectionately Rourix in honor of its inventor). In this language, 100 variables of integer type are exclusively used, $x_0 \ldots x_{99}$, all of them initially with the value 1. The instruction of the program are consecutively stored in labeled postitions $000, 001, 002, \ldots$ (at most 999). The set of possible instructions is:

`stop`	stops the program.
`endl`	writes an end of line.
`prin` i	writes, preceded of a space, the content of x_i.
`stor` $i\ v$	stores the value v in the variable x_i ($x_i := v$).
`copy` $i\ j$	copies x_j in x_i ($x_i := x_j$).
`acum` $i\ j$	accumulates x_j in x_i ($x_i := x_i + x_j$).
`subs` $i\ j$	substracts x_j of x_i ($x_i := x_i - x_j$).
`prod` $i\ j$	multiplies x_i by x_j ($x_i := x_i * x_j$).
`goto` ℓ	jumps to the instruction of the position ℓ.
`jzer` $i\ \ell$	jumps to the instruction of the position ℓ only if $x_i = 0$.
`jneg` $i\ \ell$	jumps to the instruction of the position ℓ only if $x_i < 0$.

The program always starts to execute from the position 000. After executing the instruction of a position p, it goes to the following instruction (the one in the position $p + 1$, which if has to be executed will always exist), except, obviously, with `stop`, `goto`, and `jzer` or `jneg` when the conditions to jump are true.

All the i and j of the program will be between 0 and 99. All the ℓ are valid lines of program. Executing a program, it will always find a `stop` in some moment.

Input

The input consists of a correct program in assembly language, between 1 and 1000 lines. Each line starts with three redundant digits that indicate its number starting with 000, followed by a space, and the instruction corresponding to that line. The fields of each instruction are also separated by a space. All the directions of jump ℓ have exactly three digits.

Output

Your program must print the same that would print the program in assembler.

Hint

Having a table to store the 100 variable is enough, another table to store the program, and an index that indicates which is the instruction that must be executed imediately after.

Sample input 1

```
000 prin 0
001 acum 0 0
002 prin 0
003 endl
004 stor 1 23
005 prod 0 1
006 stor 3 -9876
007 copy 4 3
008 prin 1
009 prin 0
010 prin 3
011 prin 4
012 endl
013 stop
```

Sample output 1

```
1 2
23 46 -9876 -9876
```

Sample input 2

```
000 stor 8 999
001 goto 003
002 prin 0
003 prin 8
004 endl
005 stop
006 prin 0
007 endl
008 stop
009 prin 0
```

Sample output 2

```
999
```

Sample input 3

```
000 stor 99 -10
001 prin 99
002 acum 99 0
003 jzer 99 005
004 goto 001
005 endl
006 stop
```

Sample output 3

```
-10 -9 -8 -7 -6 -5 -4 -3 -2 -1
```

Sample input 4

```
000 stor 0 95
001 stor 1 25
002 prin 0
003 prin 1
004 endl
005 subs 0 1
006 jzer 0 014
007 jneg 0 009
008 goto 002
009 acum 0 1
010 copy 2 0
011 copy 0 1
012 copy 1 2
013 goto 002
014 endl
015 prin 1
016 endl
017 stop
```

Sample output 4

```
95 25
70 25
45 25
20 25
25 20
5 20
20 5
15 5
10 5
5 5

5
```

Problem information

Author : Omer Giménez
Translator : Carlos Molina
Generation : 2014-01-29 10:35:11