

---

## Intérprete

P33564\_es

Olimpiada Informática Española — Final 2007 (2007)

---

Hacer un programa que simule la ejecución de un programa escrito en un cierto lenguaje ensamblador (Apodado cariñosamente Rourix en honor a su inventor). En este lenguaje, se dispone exclusivamente de 100 variables de tipo entero,  $x_0 \dots x_{99}$ , todas inicialmente con el valor 1. Las instrucciones del programa se encuentran guardadas consecutivamente en posiciones etiquetadas 000, 001, 002, ... (hasta un máximo de 999). El conjunto de instrucciones posibles es:

<code>stop</code>	detiene el programa.
<code>endl</code>	escribe un salto de línea.
<code>prin i</code>	escribe, precedido de un espacio, el contenido de $x_i$ .
<code>stor i v</code>	guarda el valor $v$ en la variable $x_i$ ( $x_i := v$ ).
<code>copy i j</code>	copia $x_j$ en $x_i$ ( $x_i := x_j$ ).
<code>acum i j</code>	acumula $x_j$ en $x_i$ ( $x_i := x_i + x_j$ ).
<code>subs i j</code>	resta $x_j$ de $x_i$ ( $x_i := x_i - x_j$ ).
<code>prod i j</code>	multiplica $x_i$ por $x_j$ ( $x_i := x_i * x_j$ ).
<code>goto l</code>	salta a la instrucción de la posición $l$ .
<code>jzer i l</code>	salta a la instrucción de la posición $l$ sólo si $x_i = 0$ .
<code>jneg i l</code>	salta a la instrucción de la posición $l$ sólo si $x_i < 0$ .

El programa siempre se empieza a ejecutar desde la posición 000. Después de ejecutar la instrucción de una posición  $p$ , se pasa a ejecutar la siguiente instrucción (la de la posición  $p + 1$ , la cual si se ha de ejecutar siempre existirá), excepto como es lógico con `stop`, `goto`, y `jzer` o `jneg` cuando las condiciones para saltar son ciertas.

Todas las  $i$  y  $j$  del programa estarán entre 0 y 99. Todas las  $l$  son líneas de programa válidas. Al ejecutar un programa, siempre se encontrará un `stop` en algún momento.

## Entrada

La entrada consiste en un programa correcto en ensamblador, de entre 1 y 1000 líneas. Cada línea empieza con tres dígitos redundantes que indican su número empezando por 000, seguidos de un espacio, y de la instrucción correspondiente a esa línea. Los campos de cada instrucción también están separados por un espacio. Todas las direcciones de salto  $l$  tienen exactamente tres dígitos.

## Salida

Hay que escribir lo mismo que escribiría el programa en ensamblador.

## Pista

Es suficiente con tener una tabla para almacenar las 100 variables, otra tabla para almacenar el programa, y un índice que indica cual es la instrucción que se ha de ejecutar a continuación.

### Ejemplo de entrada 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
```

### Ejemplo de salida 1

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

### Ejemplo de entrada 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
```

### Ejemplo de salida 2

```
999
```

### Ejemplo de entrada 3

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

### Ejemplo de salida 3

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

### Ejemplo de entrada 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
```

### Ejemplo de salida 4

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

## **Información del problema**

Autor : Omer Giménez

Generación : 2014-01-29 10:35:07

© *Jutge.org*, 2006–2014.

<http://www.jutge.org>