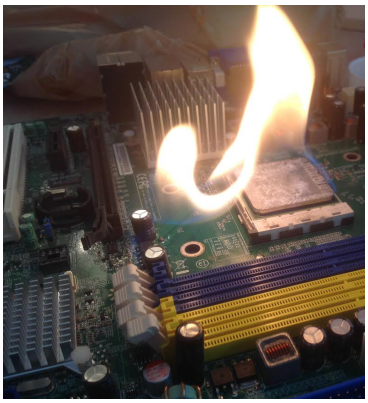
Problem H
Halt and Catch Fire
Time limit: 1 second

Research has led to the creating of a new super fast CPU called the Fast Processing Chip (FPC). This product has a never-before seen debugging function: an instruction called `hcf`, providing debugging power unmatched by any previously designed CPU.

With the product becoming popular fast, companies can't wait to get their hands on the chip to start development immediately. One problem, however, is the chip is too expensive to just hand over to every developer all willy-nilly.

You have been hired to design a program that runs as if it were a processor, so the programmers can write programs and easily test these programs without needing the hardware.



The FPC, after processing the `hcf` instruction.
(Source: https://imgur.com/gallery/zDbw6en)

Input

- One line with one integer: $1 \leq n \leq 1000$, the number of lines in the program.
- n lines with three sections, separated with spaces:
  - The operation, always 3 characters
  - An argument: Either an integer $-2^{31} \leq a \leq 2^{31} - 1$ or a variable prepended with `$`
  - An argument: Either an integer $-2^{31} \leq b \leq 2^{31} - 1$ or a variable prepended with `$`

There are five possible operations:

- `mov` – Move value from the first argument to the second
- `add` – Add value from the second argument to the first and store in `$acc`
- `sub` – Subtract value of the second argument from the first and store in `$acc`
- `jeq` – Jump to the instruction on the value of the second argument if the value of the first argument equals `$cmp`
- `hcf` – Halt and Catch Fire: Stop the program immediately, takes only integer arguments

An argument can be noted as an integer immediate or a variable reference, in which case it's prepended by `$`. All variable names consist of at most 20 lowercase characters from the English alphabet (`a-z`) or underscores (_).

There are four predefined variables:

- `$acc` – Contains the value of an `add` or `sub` operation
- `$pc` – Contains the address of the currently processed operation
- `$cmp` – Used to compare with the `jeq` operation

- `$out` – Should be output when the program finishes

Any other variables are defined as soon as a value is moved to them.
Assume any given input program to be syntactically correct.
Integers will never over- or underflow.

Output

The program exits either when the value of the `$pc` variable reaches out of the bounds of the program, or when an `hcf` instruction is reached.

- In the first case, output one line containing the value of `$out` at the end of the program.
- In the second case, output the following:
  - One line containing the `hcf` instruction that was called.
  - One line containing the content of the `$acc` variable.
  - One line containing the content of the `$cmp` variable.
  - One line containing the content of the `$out` variable.

Sample Input 1	Sample Output 1
6 mov 3 $cmp jeq 3 3 jeq $cmp 5 mov 42 $out jeq $cmp 7 mov 50 $out	42

Sample Input 2	Sample Output 2
4 mov 3 $cmp mov 2 $out add $cmp $out hcf 0 0	hcf 0 0 5 3 2