UVa Online Judge

964 Custom Language

We would like to use a custom language in the TIUP contest for writing our solutions, but we are not allowed to. However, as we like it so much, we want you to develop an interpreter for it! Our language is very simple, it works over a stack of integers and contains the following set of instructions:

`PUSH` v : puts value v on the top of the stack;

`POP` x : removes the value at the top of the stack and puts it on variable x;

`DUP` : duplicates the top of the stack, i.e, repeats the value at the top by pushing it again;

`ADD` : adds the two values at the top of the stack;

`SUB` : subtracts the two values at the top of the stack;

`MUL` : multiplies the two values at the top of the stack;

`DIV` : divides the two values at the top of the stack;

`READ` : reads a value from input and puts it on the top of the stack;

`WRITE` : writes the value at the top of the stack on output followed by a `\n`;

`JUMP` v : jumps to instruction v;

`JUMPPOS` v : jumps to instruction v, if the top of the stack is greater than 0;

`JUMPZERO` v : jumps to instruction v, if the top of the stack is 0;

A program is a list of instructions, one per line. An instruction is identified by the line number i, where $1 \leq i \leq |L|$. Whenever we reach an instruction not defined, the program ends. For noncommutative operations, the top of the stack is the first argument. A variable s is a string in {`a`, ..., `z`}.{`1`, ..., `9`, `a`, ..., `z`}* such that $s \leq 100$. In the above, v can be a constant or a variable. All arithmetic operations remove their arguments from the top of the stack and put their output on the top of the stack. Finally, when any instruction reads the top of the stack it removes it.

Input

Several programs, each consisting of a code section and a data section.

A code section consists of a list of instructions, one per line, and is terminated by a line with the symbol '`#`'. This is followed by the a data section consisting of a set of integers, one per line, which are the inputs for the program. A data section is also terminated by a line with the symbol '`#`'.

Each program has one code section and one data section; a data section can be empty.

Output

For each program the output obtained by the program for the given input (if any) or '`ABORTED`' if something wrong happens. The output of each program should be terminiated by a line with the symbol '`#`'.

Sample Input

```
READ
POP num
PUSH 2
POP x
PUSH num
PUSH 2
SUB
JUMPPOS 31
PUSH num
PUSH x
SUB
DUP
JUMPZERO 28
JUMPPOS 28
PUSH x
PUSH num
DIV
PUSH x
MUL
PUSH num
SUB
JUMPZERO 31
PUSH x
PUSH 1
ADD
POP x
JUMP 9
PUSH 1
WRITE
JUMP 33
PUSH 0
WRITE
#
7
#
PUSH undefined
WRITE
#
#
```

Sample Output

```
1
#
ABORTED
#
```