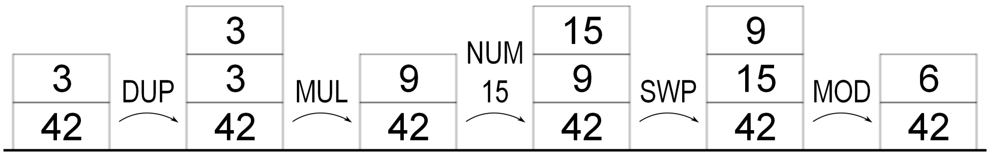
# Stack Machine Executor

`execute.c`, `execute.C`, `execute.java`

Many ciphers can be computed much faster using various machines and automata. In this problem, we will focus on one particular type of machines called *stack machine*. Its name comes from the fact that the machine operates with the well-known data structure — *stack*. The later-stored values are on the top, older values at the bottom. Machine instructions typically manipulate the top of the stack only.

Our stack machine is relatively simple: It works with integer numbers only, it has no storage beside the stack (no registers etc.) and no special input or output devices. The set of instructions is as follows:

- **NUM *X***, where $X$ is a non-negative integer number, $0 \le X \le 10^9$. The NUM instruction stores the number $X$ on top of the stack. It is the only parametrized instruction.
- **POP**: removes the top number from the stack.
- **INV**: changes the sign of the top-most number. ($42 \to -42$)
- **DUP**: duplicates the top-most number on the stack.
- **SWP**: swaps (exchanges) the position of two top-most numbers.
- **ADD**: adds two numbers on the top of the stack.
- **SUB**: subtracts the top-most number from the "second one" (the one below).
- **MUL**: multiplies two numbers on the top of the stack.
- **DIV**: integer divison of two numbers on the top. The top-most number becomes divisor, the one below dividend. The quotient will be stored as the result.
- **MOD**: modulo operation. The operands are the same as for the division but the remainder is stored as the result.




All binary operations consider the top-most number to be the "right" operand, the second number the "left" one. All of them remove both operands from the stack and place the result on top in place of the original numbers.

If there are not enough numbers on the stack for an instruction (one or two), the execution of such an instruction will result into a program *failure*. A failure also occurs if a divisor becomes zero (for DIV or MOD) or if the result of any operation should be more than $10^9$ in absolute value. This means that the machine only operates with numbers between $-1\,000\,000\,000$ and $1\,000\,000\,000$, inclusive.

To avoid ambiguities while working with negative divisors and remainders: If some operand of a division operation is negative, the absolute value of the result should always be computed with absolute values of operands, and the sign is determined as follows: The quotient is negative if (and only if) exactly one of the operands is negative. The remainder has the same sign as the dividend. Thus, 13 div $-4 = -3$, $-13$ mod $4 = -1$, $-13$ mod $-4 = -1$, etc.

If a failure occurs for any reason, the machine stops the execution of the current program and no other instructions are evaluated in that program run.

## Input Specification

The input contains description of several machines. Each machine is described by two parts: the program and the input section.

The program is given by a series of instructions, one per line. Every instruction is given by three uppercase letters and there must not be any other characters. The only exception is the NUM instruction, which has exactly one space after the three letters followed by a non-negative integer number between 0 and $10^9$. The only allowed instructions are those defined above. Each program is terminated by a line containing the word "`END`" (and nothing else).

The input section starts with an integer $N$ ($0 \le N \le 10\,000$), the number of program executions. The next $N$ lines contain one number each, specifying an input value $V_i$, $0 \le V_i \le 10^9$. The program should be executed once for each of these values independently, every execution starting with the stack containing one number — the input value $V_i$.

There is one empty line at the and of each machine description. The last machine is followed by a line containing the word "`QUIT`". No program will contain more than 100 000 instructions and no program requires more than 1 000 numbers on the stack in any moment during its execution.

## Output Specification

For each input value, print one line containing the output value for the corresponding execution, i.e., the one number that will be on the stack after the program executes with the initial stack containing only the input number.

If there is a program failure during the execution or if the stack size is incorrect at the end of the run (either empty or there are more numbers than one), print the word "`ERROR`" instead.

Print one empty line after each machine, including the last one.

## Sample Input

```
DUP
MUL
NUM 2
ADD
END
3
1
10
50

NUM 1
NUM 1
ADD
END
2
42
43
NUM 600000000
ADD
END
3
0
600000000
1

QUIT
```

## Output for Sample Input

```
3
102
2502

ERROR
ERROR

600000000
ERROR
600000001
```