CCO '03 P1 - BFed

Time limit: 1.0s **Memory limit:** 16M

Canadian Computing Competition: 2003 Stage 2, Day 1, Problem 1

For this problem, you will write an interpreter for an extremely simple programming language. The language does have a name, but for various, ahem, reasons, we shall call it BF.

A BF program operates on a simple 1-dimensional array of memory cells; there is a pointer to a "current" memory cell. In "vanilla" BF, this array has a size of $30\,000$, and each cell is an 8-bit integer - that is, cells store values in the range of 0 -255. Incrementing a cell with a value of 255 wraps around to 0; decrementing a cell with a value of 0 wraps around to 255. All cells are initially set to 0, and the pointer initially points to the leftmost cell.

The BF program consists of a string. Each character can be one of 8 "commands":

Command	Description
`>`	move the pointer right one cell
`<`	move the pointer left one cell
`+`	increment the current cell by 1
`-`	decrement the current cell by 1
`[`	skips ahead to the matching `]` IF the current cell contains 0
`]`	returns to the matching `[` UNLESS the current cell contains 0
`.`	outputs the current cell as a character
`,`	inputs one character into the current cell

You do not need to implement the `,` command.

You should ignore any characters in the BF program except in the first 7 commands listed above. The program ends when there are no more characters to be processed.

Interestingly enough, these commands are powerful enough that a BF program can (given sufficient memory, time and programming patience) perform any computation that any other programming language can do!

Input Specification

Your interpreter will be given a BF program in standard input. It may span multiple lines. The program will be terminated by a hash mark `#`.

You may assume that no programs will be given to your interpreter that are invalid, run unreasonably long (or forever), or crash off the left or right end of the array. No input will be longer than $10\,000$ characters.

Output Specification

Your interpreter should print the output from the execution of the BF program. Do not print any characters other than the ones from the program.

Sample Input 1

```
++[>+++++++++++++<-]          // put 26 in cell 1
>>>++++++++[<+++++++++>-]<+   // put 65 in cell 2
<[->.+<]                      // output alphabet
++++++++++.                   // output newline
#
```

Sample Output 1

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

Sample Input 2

```
[+[>uh-oh<]+]------outer[+>------inter[+>-----<]<]>>.<++++++++++.#
```

Sample Output 2

```
L
```

Sample Input 3

```
>+[>[--------->]---[<]>+++]>>>>>[-]>>>.>>>++.>..>------.[<]<
----.>>>>>>---.>>>.+++.<.<-.[<]<+.<<<+.#
```

Sample Output 3

```
Hello World!
```