Brainfuck (not capitalized except at the start of a sentence) is an extremely minimal Turing-complete programming language with just 8 commands.

You can try brainfuck on your browser with brainfuck-visualizer.

```
Any character not "><+-.,[]" (excluding quotation marks) is ignored.

Brainfuck is represented by an array with 30,000 cells initialized to zero
and a data pointer pointing at the current cell.

There are eight commands:
+ : Increments the value at the current cell by one.
- : Decrements the value at the current cell by one.
> : Moves the data pointer to the next cell (cell on the right).
< : Moves the data pointer to the previous cell (cell on the left).
. : Prints the ASCII value at the current cell (i.e. 65 = 'A').
, : Reads a single input character into the current cell.
[ : If the value at the current cell is zero, skips to the corresponding ] .
    Otherwise, move to the next instruction.
] : If the value at the current cell is zero, move to the next instruction.
    Otherwise, move backwards in the instructions to the corresponding [ .

[ and ] form a while loop. Obviously, they must be balanced.

Let's look at some basic brainfuck programs.

++++++ [ > ++++++++++ < - ] > +++++ .

This program prints out the letter 'A'. First, it increments cell #1 to 6.
Cell #1 will be used for looping. Then, it enters the loop ([) and moves
to cell #2. It increments cell #2 10 times, moves back to cell #1, and
decrements cell #1. This loop happens 6 times (it takes 6 decrements for
cell #1 to reach 0, at which point it skips to the corresponding ] and
continues on).

At this point, we're on cell #1, which has a value of 0, while cell #2 has a
value of 60. We move on cell #2, increment 5 times, for a value of 65, and then
print cell #2's value. 65 is 'A' in ASCII, so 'A' is printed to the terminal.


, [ > + < - ] > .

This program reads a character from the user input and copies the character into
cell #1. Then we start a loop. Move to cell #2, increment the value at cell #2,
move back to cell #1, and decrement the value at cell #1. This continues on
until cell #1 is 0, and cell #2 holds cell #1's old value. Because we're on
cell #1 at the end of the loop, move to cell #2, and then print out the value
in ASCII.

Also keep in mind that the spaces are purely for readability purposes. You
could just as easily write it as:

,[>+<-]>.

Try and figure out what this program does:
```

```
,>,< [ > [ >+ >+ << -] >> [- << + >>] <<< -] >>

This program takes two numbers for input, and multiplies them.

The gist is it first reads in two inputs. Then it starts the outer loop,
conditioned on cell #1. Then it moves to cell #2, and starts the inner
loop conditioned on cell #2, incrementing cell #3. However, there comes a
problem: At the end of the inner loop, cell #2 is zero. In that case,
inner loop won't work anymore since next time. To solve this problem,
we also increment cell #4, and then recopy cell #4 into cell #2.
Then cell #3 is the result.
```

And that's brainfuck. Not that hard, eh? For fun, you can write your own brainfuck programs, or you can write a brainfuck interpreter in another language. The interpreter is fairly simple to implement, but if you're a masochist, try writing a brainfuck interpreter… in brainfuck.