The Encrypted Wisdom

While cataloging a long-forgotten archive of early computing experiments, a curious discovery was made: a fragment of code written in a strange and cryptic language, unlike anything seen before. The paper it was found on was brittle and aged, suggesting it had been hidden away for decades. Scattered notes found nearby suggest that this code was not merely written; it was carefully crafted to be deliberately difficult to read, a challenge left behind for future generations. Every symbol seems chosen to obscure rather than reveal, turning the simple act of reading into a true test of determination. Despite its bewildering appearance, experts believe it holds a piece of timeless wisdom for developers, a message the original author intended only the most persistent minds to uncover. Here is the recovered code:

```
0a2<<<*f+444**8+be2+*E+B84|4+*1+22*<<<e+85a^1+*e+BD55*9-*9+482**8+50~F&1+*6+B8<*
b+87F&<2+*C+9C2^2+*1+7<<4*8+e93*B-*8+8B5+*5a6+*1+e2C*8-*2+8<4*<F+64<<*7+18<*d+ac
4|4+*5+7F1+*c88+*A+Dfd^<<<*1+537*5-*9+964*8-*8+6F1+*E+5b2*6-*5+dC4+*d+C71<<<<*f+
9d3+*A+a8*<5+9A6+*A+897+*2+8Ff^~F&1+*D+A3<5<+*d+E1f+*9+797*5%d+*8+a65*7%E+*e+188
+*<<<<88<<2/*1+b2<<<<F44**8+De2+*B+184|4+*4+82*<<<4+E5a^1+*3+855*9-*e+a82**4+10~
f&1+*B+28<*9+17f&<2+*E+8c2^2+*e+6<<4*A+793*b-*1+cb5+*fa6+*4+d2C*8-*6+2<4*<1+24<<
*6+78<*1+Dc4|4+*c+bf1+*7+988+*2+FFD^<<<*9+B37*5-*764*8-*9+5f1+*6+8b2*6-*9+5C4+*C
+C1<<<<*5+9d3+*e8*<a+Ea6+*497+*7+Fff^~F&1+*6+33<5<+*5+21f+*2+197*5%d+*e+a65*7%e+
*2+C88+*6+3<<<<d+18<<2/*A+e2<<<*4+C44**9E2+*5+9C84|4+*2+d2*<<<C+b5A^1+*3+c55*9-*
1+c82**6+d0~f&1+*6+78<*2+C7f&<2+*9+6C2^2+*8+1<<4*c+493*b-*e+9B5+*9+6A6+*e+62C*8-
*F+f<4*<4+74<<*c+a8<*A+47c4|4+*2F1+*7+d88+*6+dfd^<<<*d+137*5-*B64*8-*4+8F1+*a+6b
2*6-*2+5C4+*d+71<<<<*6+54d3+*A+A8*<4+2A6+*d+297+*2+7Ff^~f&1+*4+93<5<+*c+61F+*7+B
97*5%d+*a65*7%e+*E+888+*d+5<<<<1+98<<2/*7+72<<<*7+244**4+bE2+*F+184|4+*5+d2*<<<A
+55a^1+*e+c55*9-*3+D82**6+b40~f&1+*B+18<*7+F7f&<2+*e+7C2^2+*6+f<<4*E+993*b-*5+4b
5+*2+4a6+*7+12C*8-*<4*<D4<<*4+28<*3+Fc4|4+*D+75F1+*E+D88+*9+1FD^<<<*2+F37*5-*864
*8-*6+99f1+*2+Bb2*6-*8+92c4+*1+51<<<<*7+1d3+*2+48*<6+2a6+*1+397+*9+Bff^~f&1+*4+4
3<5<+*71F+*1297*5%d+*c+265*7%E+*5+588+*E+9<<<<B+58<<2/*b+32<<<*1+444**9+6e2+*8+8
84|4+*7+B2*<<<6+95a^1+*1+255*9-*e82**3+f0~F&1+*A+28<*2+77F&<2+*F+5C2^2+*b<<4*c+d
93*B-*2+4<<4*9+.$688+*E+.$7<4*<.$764*8-*5+.$71<<<<*4+.$2A6+*.$63<5<+*b+.$68<*5+.
$7B2*6-*9+.$2fD^<<<*.$34<<*1+.$31F+*A+.$288+*.$?465*7%e+*9+.$60~f&1+*E+.$75a^1+*
.$78*<5+.$784|4+*4+.$22c*8-*.$68<<2/*B+.$6A6+*5+.$7C4|4+*9+.$2ff^~F&1+*.$32*<<<2
+.$32<<<*A+.$28<*.$?@:#:@@1+#*544**d+2*<<<9+97*5%d+*5+D3+*F+A6+*3+<<<<5+2<<<*1+C
4|4+*e+3<5<+*4+7F&<2+*4+93*B-*e+<4*<a+5A^1+*C+2c*8-*^=($@[1-##5<<<<E+1<<<<*3+fd^
<<<*8+8*<5+97*5%d+*B+*144**88+*7+f1+*9+C2^2+*7+1f+*2C*8-*3+8<*6+a6+*1+*80~f&1+*7
f&<2+*2<<<*B2*6-*93*b-*C4|4+*2*<<<%@255*9-*A+<4*<3+65*7%E+*1+e2+*7+f1+*5+*764*8-
*8+97+*5a^1+*9+37*5-*C+<<4*5+3<5<+*4+84|4+*D++882**Ff^~f&1+*4<<*8<<2/*88+*8<*A6+
*%:@1D3+*c4+*%^##:@1b5+*<<<<%@:#:@^:#^@^^.$]A.$=)=(4<4*<9+.$61<<<<*E+.$7F1+*6+.$
6b5+*1+.$6fd^<<<*C+.$68<*9+.$6a6+*4+.$237*5-*.$64<<*B+.$6<<<<5+.$784|4+*9+.$78<<
2/*3+.$2C4+*c+.$28<*.$7c4|4+*4+.$755*9-*2+.$73<5<+*9+.$2FF^~F&1+*.$61f+*1+.$693*
b-*7+.$6e2+*1+.$67F&<2+*9+.$697*5%D+*E+.$2C2^2+*e+.$a.$)$de2+*4+82**3+97*5%d+*1+
```

At first glance, it might seem like an impossible challenge. Fortunately, alongside the code, we also uncovered a set of documents that appear to describe the language's specifications, as well as a mysterious piece of paper containing two cryptic numbers. Here they are:

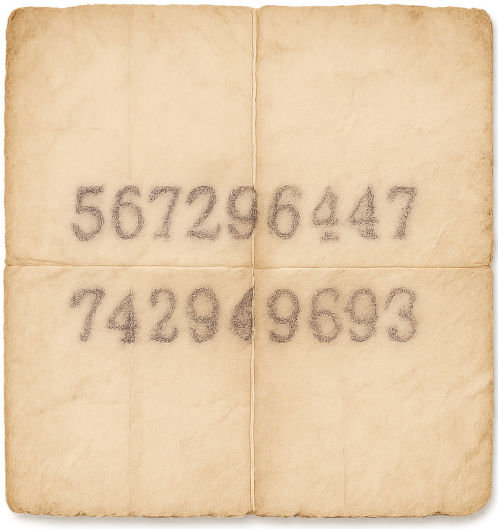

Figure 1: Mysterious piece of paper

Language Specification

The interpreter should process one character at a time, ignoring any character that is not explicitly defined below. It operates using a data model that consists of two stacks: a working stack and an auxiliary stack, both initially empty. All values stored in the stacks should be treated as at least 64-bit signed integers.

Literals

Any hexadecimal digit case-insensitive (`0-9`, `A-F`, `a-f`), pushes its numeric value ($0 \ldots 15$) onto the *working* stack.

Stack Manipulation

`:`	Duplicate top of working stack
`$`	Pop and discard top of working stack
`@`	Pop top of working stack, push onto auxiliary stack
`#`	Pop top of auxiliary stack, push onto working stack

Input/Output

`?`	Read *one* integer from `stdin` and push it
`!`	Print the *top* of the working stack in decimal
`.`	Print the *top* of the working stack as an ASCII character

Unary Operations

Each unary operation must pop one value from the working stack, apply the operation, and push the result back onto the working stack.

`~`	Bit-wise NOT of top value
`<`	Left-shift top value by 1 bit
`>`	Right-shift top value by 1 bit

Binary Operations

Each binary operation pops two values from the working stack (first b, then a), applies the operation $a \circ b$, and pushes the result back onto the working stack. .

`+`	Addition
`-`	Subtraction
`*`	Multiplication
`/`	Integer division (floor toward zero)
`%`	Modulo
`&`	Bit-wise AND
`|`	Bit-wise OR
`^`	Bit-wise XOR

Zero/One Conversion

`=`	Pop one value from the working stack; push 0 if > 0, else push 1.

Control Flow

- **Loops** `[ ... ]`
  On `[` , if top of **working** = 0, jump forward to the matching `]`. On `]`, if the top is ≠ 0, jump back to the matching `[`.
- **Conditionals** `( ... )`
  On `(` , if top of **working** = 0, jump forward to the matching `]`. On `)`, execution continues normally *no backward jump*.

Examples

Prints "`hello world`":

```
6 < < < < 8 + . 3 - . 7 + . . 3 + . > > 5 + . < < 9 - . 8 - . 3 + . 6 - . 8 - . $ a . $
```

Read a number and print its square; repeat until 0 is entered.

```
? [ : * ! $ a . $ ? ]
```

Read an integer N, then print the first N numbers of the Fibonacci sequence:

```
? 0 @ 1 @ [ # : # ! + @ : # : @ ^ : # ^ @ ^ @ 1 - ( f 3 * 1 - . c - . $ ) ] a . $
```

Read an integer N, then print all its divisors, one per line:

```
1 @ ? [ : # : @ % = ( # ! a . $ @ ) $ : 1 + # 1 + : @ - = ( 1 ) = ( $ ) ]
```