KaoLang (👉˚ワ˚)👉 Specification

created by Bao Vuong, 6/2/2025

KaoLang (👉˚ワ˚)👉 is an esoteric programming language inspired by *Brainf*ck*. It is written and maintained by Bao Vuong (aka Vbee). KaoLang (👉˚ワ˚)👉 exists due to the lack of kaomoji usage in esoteric languages. Don't get tricked by the naive look of KaoLang (👉˚ワ˚)👉 because it forces the user to do bit twiddling instead of + or - operation from *Brainf*ck*.

Commands

Kaomoji	Functionality
👈(˚ワ˚👈)	Shift current memory cell value **left** by 1 (multiply by 2)
(👉˚ワ˚)👉	Shift current memory cell value **right** by 1 (integer divide by 2)
👈(⌒▽⌒)👉	Apply **NAND** between current memory cell and value in register; store result in current memory cell; sets register value to 0 after operation
(へ･_･)へ┳━┳	Begin loop if current memory cell is **not zero**
(ノ °□°) ノ ⁀ ┻━━┻	End loop — jump back to matching loop start if current cell is **not zero**
✍(◔◡◔)	Output the character corresponding to the current memory cell
(❶_❶)	Read one byte from input into the current memory cell
o((>ω<))o< td>	Move memory pointer **right** by 1 cell
o((>ω<))o	Move memory pointer **left** by 1 cell
ᑫ(ò_óˇ)ᕤ	Copy current memory cell value into the **register** if register is empty, else replace memory cell value with value in register + set register to empty

Computational Class

KaoLang (👉˚ワ˚)👉 is Turing complete because every command can be mapped to a *Brainf*ck* command.

This table maps standard Brainf*ck commands to their corresponding implementations in KaomojiLang, proving its Turing completeness.

Brainf*ck	KaoLang (👉˚ワ˚)👉 Equivalent	Description
>	o((>ω<))o	Move memory pointer right

Brainf*ck	KaoLang (👉゜ワ゜)👉 Equivalent	Description
<	o((>ω<))o	Move memory pointer left
+	(see below)	Increment current memory cell by 1 using NAND and shifts
−	(see below)	Decrement current memory cell by 1 using NAND and shifts
.	✍(◔◡◔)	Output the character at current memory cell
,	(⓿_⓿)	Read one byte into current memory cell
[	(へ･_･)へ┳┳	Start loop if current cell is not 0
]	(╯°□°）╯︵ ┻┻	Jump to matching [if current cell is not 0

+ — Increment Implementation

```
cell[0] stores a (original number); cell[1] stores b (= 1);
o(( >ω<))o                    // Move to cell[1]
👈(⌒▽⌒)👉                      // NAND with self → 255
ᗡ(ò_óˇ)Ↄ                      // Copy 255 to register
(👉゜ワ゜)👉                    // Shift right → 127
👈(゜ワ゜👈)                    // Shift left → 254
👈(⌒▽⌒)👉                      // NAND(255, 254) = 1; cell[1] = 1; register
reset

(へ･_･)へ┳┳                   // While b ≠ 0
o((>ω< ))o ᗡ(ò_óˇ)Ↄ           // Copy a to register
o(( >ω<))o o(( >ω<))o ᗡ(ò_óˇ)Ↄ // Paste a to cell[2]
o((>ω< ))o ᗡ(ò_óˇ)Ↄ           // Copy b
o(( >ω<))o 👈(⌒▽⌒)👉           // NAND a b; register reset
ᗡ(ò_óˇ)Ↄ                      // Copy NAND result to reg
👈(⌒▽⌒)👉                      // NAND again to get a & b; register reset

o((>ω< ))o ᗡ(ò_óˇ)Ↄ           // Copy b to reg
o(( >ω<))o o(( >ω<))o ᗡ(ò_óˇ)Ↄ // Paste b to cell[3]
ᗡ(ò_óˇ)Ↄ 👈(⌒▽⌒)👉             // ~b in cell[3]; register reset

o((>ω< ))o o((>ω< ))o o((>ω< ))o ᗡ(ò_óˇ)Ↄ  // Copy a
o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o ᗡ(ò_óˇ)Ↄ // Paste a to cell[4]
ᗡ(ò_óˇ)Ↄ 👈(⌒▽⌒)👉             // ~a in cell[4]; register reset

o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o ᗡ(ò_óˇ)Ↄ// Back to a, copy a
o(( >ω<))o o(( >ω<))o o(( >ω<))o  // To ~b
👈(⌒▽⌒)👉                        // cell[3] = NAND a ~b; register reset

o((>ω< ))o o((>ω< ))o ᗡ(ò_óˇ)Ↄ // Copy b
o(( >ω<))o o(( >ω<))o o(( >ω<))o // To ~a
👈(⌒▽⌒)👉                        // cell[4] = NAND b ~a; register reset
```

```
ᕦ(ò_óˇ)ᕤ o((>ω< ))o 👈 (⌒▽⌒)👉  // Final XOR: (a NAND ~b) NAND (b NAND ~a); register reset

ᕦ(ò_óˇ)ᕤ o((>ω< ))o o((>ω< ))o o((>ω< ))o ᕦ(ò_óˇ)ᕤ // Paste XOR result back to cell a

o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ // Copy carry
o((>ω< ))o ᕦ(ò_óˇ)ᕤ            // Move to b and paste carry
👈 (ﾟヮﾟ👈 )                    // b = carry << 1

(╯°□°)╯︵ ┻━┻                  // End loop
o((>ω< ))o // return to original value that has been incremented
```

— Decrement Implementation

```
cell[0] stores a (original number); cell[1] stores b (= 255);
o(( >ω<))o                   // Move to cell[1]
👈 (⌒▽⌒)👉                    // NAND with self → 255

(ノ・_・)ノ┳━┳                  // While b ≠ 0
o((>ω< ))o ᕦ(ò_óˇ)ᕤ           // Copy a to register
o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ // Paste a to cell[2]
o((>ω< ))o ᕦ(ò_óˇ)ᕤ           // Copy b
o(( >ω<))o 👈 (⌒▽⌒)👉          // NAND a b; register reset
ᕦ(ò_óˇ)ᕤ                      // Copy NAND result to reg
👈 (⌒▽⌒)👉                     // NAND again to get a & b; register reset

o((>ω< ))o ᕦ(ò_óˇ)ᕤ           // Copy b to reg
o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ // Paste b to cell[3]
ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉             // ~b in cell[3]; register reset

o((>ω< ))o o((>ω< ))o o((>ω< ))o ᕦ(ò_óˇ)ᕤ  // Copy a
o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ // Paste a to cell[4]
ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉             // ~a in cell[4]; register reset

o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o ᕦ(ò_óˇ)ᕤ// Back to a, copy a
o(( >ω<))o o(( >ω<))o o(( >ω<))o  // To ~b
👈 (⌒▽⌒)👉                         // cell[3] = NAND a ~b; register reset

o((>ω< ))o o((>ω< ))o ᕦ(ò_óˇ)ᕤ // Copy b
o(( >ω<))o o(( >ω<))o o(( >ω<))o // To ~a
👈 (⌒▽⌒)👉                         // cell[4] = NAND b ~a; register reset

ᕦ(ò_óˇ)ᕤ o((>ω< ))o 👈 (⌒▽⌒)👉  // Final XOR: (a NAND ~b) NAND (b NAND ~a); register reset

ᕦ(ò_óˇ)ᕤ o((>ω< ))o o((>ω< ))o o((>ω< ))o ᕦ(ò_óˇ)ᕤ // Paste XOR result back to cell a
```

```
o(( >ω<))o o(( >ω<))o ᘿ(ò_óˇ)ᕤ // Copy carry
o((>ω< ))o ᘿ(ò_óˇ)ᕤ             // Move to b and paste carry
👈(ﾟヮﾟ👈)                      // b = carry << 1

(╯°□°)╯︵ ┻━┻                  // End loop
o((>ω< ))o // return to original value that has been decremented
```

Example Programs

XKCD Random Number

```
the next kaomoji (NAND) cannot be the start of the program because the
unicode will automatically change, so here are some words👈(⌒▽⌒)👉👈(ﾟヮﾟ
👈)👈(ﾟヮﾟ👈)👈(ﾟヮﾟ👈)👈(ﾟヮﾟ👈)👈(ﾟヮﾟ👈)👈(ﾟヮﾟ👈)(👉ﾟヮﾟ)👉 (👉ﾟ
ヮﾟ)👉 ᘿ(ò_óˇ)ᕤ 👈(⌒▽⌒)👉 // cell[1] = !(00110000)
o(( >ω<))o 👈(⌒▽⌒)👉 (👉ﾟヮﾟ)👉 ᘿ(ò_óˇ)ᕤ 👈(⌒▽⌒)👉 👈(ﾟヮﾟ👈)👈(ﾟヮﾟ👈)
ᘿ(ò_óˇ)ᕤ 👈(⌒▽⌒)👉 // cell[2] = !(00000100)
ᘿ(ò_óˇ)ᕤ o((>ω< ))o 👈(⌒▽⌒)👉 ✍(◔◡◔) // print '4' (ascii 52)
```

Hello, World!

```
cell[0] = location where we work out the characters; cell[1] - cell[8]:
the according NOT presentation of the bit so that we can use them in OR =
notA NAND notB (notA = A NAND A)
o(( >ω<))o                    // Move to cell [1]
👈(⌒▽⌒)👉                     // Apply NAND to self → 255
👈(ﾟヮﾟ👈)👈(ﾟヮﾟ👈)👈(ﾟヮﾟ👈)👈(ﾟヮﾟ👈)👈(ﾟヮﾟ👈)👈(ﾟヮﾟ👈)👈(ﾟヮﾟ👈)
// Shift left 7 times, cell[1] = 128
ᘿ(ò_óˇ)ᕤ o(( >ω<))o (👉ﾟヮﾟ)👉 // Copy current bit, move to next cell,
shift right to /2, cell[2] = 64
ᘿ(ò_óˇ)ᕤ o(( >ω<))o (👉ﾟヮﾟ)👉 // Copy current bit, move to next cell,
shift right to /2, cell[3]= 32
ᘿ(ò_óˇ)ᕤ o(( >ω<))o (👉ﾟヮﾟ)👉 // Copy current bit, move to next cell,
shift right to /2, cell[4] = 16
ᘿ(ò_óˇ)ᕤ o(( >ω<))o (👉ﾟヮﾟ)👉 // Copy current bit, move to next cell,
shift right to /2, cell[5] = 8
ᘿ(ò_óˇ)ᕤ o(( >ω<))o (👉ﾟヮﾟ)👉 // Copy current bit, move to next cell,
shift right to /2, cell[6] = 4
ᘿ(ò_óˇ)ᕤ o(( >ω<))o (👉ﾟヮﾟ)👉 // Copy current bit, move to next cell,
shift right to /2, cell[7] = 2
ᘿ(ò_óˇ)ᕤ o(( >ω<))o (👉ﾟヮﾟ)👉 // Copy current bit, move to next cell,
shift right to /2, cell[8] = 1

set every cell[1] - cell[8] to NOT(cell[x])
ᘿ(ò_óˇ)ᕤ 👈(⌒▽⌒)👉 o((>ω< ))o // Copy value to register, NAND with self,
move to next cell
ᘿ(ò_óˇ)ᕤ 👈(⌒▽⌒)👉 o((>ω< ))o // Copy value to register, NAND with self,
move to next cell
```

```
ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  o((>ω< ))o // Copy value to register, NAND with self, move to next cell
ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  o((>ω< ))o // Copy value to register, NAND with self, move to next cell
ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  o((>ω< ))o // Copy value to register, NAND with self, move to next cell
ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  o((>ω< ))o // Copy value to register, NAND with self, move to next cell
ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  o((>ω< ))o // Copy value to register, NAND with self, move to next cell
ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  // Copy value to register, NAND with self, move to next cell

o((>ω< ))o // move back to cell[0]

'H': 72 = cell[2] + cell[5]
ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  // NAND cell[0]
o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ // copy cell[2]
o((>ω< ))o o((>ω< ))o 👈 (⌒▽⌒)👉  // NAND cell[0] cell[2]

ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  // NAND cell[0]
o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ // copy cell[5]
o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o 👈 (⌒▽⌒)👉  // NAND cell[0] cell[5]
✍(◔◡◔) // Print 'H'

reset cell[0]
👈 (⌒▽⌒)👉  -> 255
ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  -> 0

'e': 101 = cell[2] + cell[3] + cell[6] + cell[8]
ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  // NAND cell[0]
o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ // copy cell[2]
o((>ω< ))o o((>ω< ))o 👈 (⌒▽⌒)👉  // NAND cell[0] cell[2]

ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  // NAND cell[0]
o(( >ω<))o o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ // copy cell[3]
o((>ω< ))o o((>ω< ))o o((>ω< ))o 👈 (⌒▽⌒)👉  // NAND cell[0] cell[3]

ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  // NAND cell[0]
o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ // copy cell[6]
o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o 👈 (⌒▽⌒)👉  // NAND cell[0] cell[6]

ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  // NAND cell[0]
o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ // copy cell[8]
o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o 👈 (⌒▽⌒)👉  // NAND cell[0] cell[8]

✍(◔◡◔) // Print 'e'
```

```
reset cell[0]
👈 (⌒▽⌒)👉  -> 255
ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  -> 0

// 'l': 108 = cell[2] + cell[3] + cell[5] + cell[6]
ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  // NAND cell[0]
o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ // copy cell[2]
o((>ω< ))o o((>ω< ))o 👈 (⌒▽⌒)👉  // NAND cell[0] cell[2]

ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  // NAND cell[0]
o(( >ω<))o o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ // copy cell[3]
o((>ω< ))o o((>ω< ))o o((>ω< ))o 👈 (⌒▽⌒)👉  // NAND cell[0] cell[3]

ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  // NAND cell[0]
o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ // copy cell[5]
o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o 👈 (⌒▽⌒)👉  // NAND cell[0] cell[5]

ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  // NAND cell[0]
o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ // copy cell[6]
o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o 👈 (⌒▽⌒)👉  // NAND cell[0] cell[6]
✍(◔◡◔) // Print 'l'
✍(◔◡◔) // Print 'l' one more time

// 'o': 111 = cell[2] + cell[3] + cell[5] + cell[6] + cell[7] + cell[8]
ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  // NAND cell[0]
o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ // copy cell[7]
o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o 👈 (⌒▽⌒)👉  // NAND cell[0] cell[7]

ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  // NAND cell[0]
o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ // copy cell[8]
o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o 👈 (⌒▽⌒)👉  // NAND cell[0] cell[8]
✍(◔◡◔) // Print 'o'

// ',': 44 = cell[3] + cell[5] + cell[6]
ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  // NAND cell[0]
o(( >ω<))o o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ // copy cell[3]
o((>ω< ))o o((>ω< ))o o((>ω< ))o 👈 (⌒▽⌒)👉  // NAND cell[0] cell[3]
ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  // NAND cell[0]
o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ // copy cell[5]
o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o 👈 (⌒▽⌒)👉  // NAND cell[0] cell[5]
ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  // NAND cell[0]
o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ // copy cell[6]
o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o 👈 (⌒▽⌒)
```

```
👉  // NAND cell[0] cell[6]
✍(◔◡◔) // Print ','
👈 (⌒▽⌒)👉  -> 255
ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  -> 0

// ' ': 32 = cell[3]
ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  // NAND cell[0]
o(( >ω<))o o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ // copy cell[3]
o((>ω< ))o o((>ω< ))o o((>ω< ))o 👈 (⌒▽⌒)👉  // NAND cell[0] cell[3]
✍(◔◡◔) // Print ' '
👈 (⌒▽⌒)👉  -> 255
ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  -> 0

// 'W': 87 = cell[2] + cell[4] + cell[6] + cell[7] + cell[8]
ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  // NAND cell[0]
o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ // copy cell[2]
o((>ω< ))o o((>ω< ))o 👈 (⌒▽⌒)👉  // NAND cell[0] cell[2]
ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  // NAND cell[0]
o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ // copy cell[4]
o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o 👈 (⌒▽⌒)👉  // NAND cell[0] cell[4]
ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  // NAND cell[0]
o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ // copy cell[6]
o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o 👈 (⌒▽⌒)👉  // NAND cell[0] cell[6]
ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  // NAND cell[0]
o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ // copy cell[7]
o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o 👈 (⌒▽⌒)👉  // NAND cell[0] cell[7]
ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  // NAND cell[0]
o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ // copy cell[8]
o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o 👈 (⌒▽⌒)👉  // NAND cell[0] cell[8]
✍(◔◡◔) // Print 'W'
👈 (⌒▽⌒)👉  -> 255
ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  -> 0

// 'o': 111 = cell[2] + cell[3] + cell[5] + cell[6] + cell[7] + cell[8]
ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  // NAND cell[0]
o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ // copy cell[2]
o((>ω< ))o o((>ω< ))o 👈 (⌒▽⌒)👉  // NAND cell[0] cell[2]
ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  // NAND cell[0]
o(( >ω<))o o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ // copy cell[3]
o((>ω< ))o o((>ω< ))o o((>ω< ))o 👈 (⌒▽⌒)👉  // NAND cell[0] cell[3]
ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  // NAND cell[0]
o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ // copy cell[5]
o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o 👈 (⌒▽⌒)👉  // NAND cell[0] cell[5]
ᕦ(ò_óˇ)ᕤ 👈 (⌒▽⌒)👉  // NAND cell[0]
o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ
```

```
// copy cell[6]
o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o 👈(⌒▽⌒)
👉 // NAND cell[0] cell[6]
Ꮚ(ò_óˇ)Ↄ 👈(⌒▽⌒)👉 // NAND cell[0]
o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o((
>ω<))o Ꮚ(ò_óˇ)Ↄ // copy cell[7]
o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω<
))o 👈(⌒▽⌒)👉 // NAND cell[0] cell[7]
Ꮚ(ò_óˇ)Ↄ 👈(⌒▽⌒)👉 // NAND cell[0]
o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o((
>ω<))o o(( >ω<))o Ꮚ(ò_óˇ)Ↄ // copy cell[8]
o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω<
))o o((>ω< ))o 👈(⌒▽⌒)👉 // NAND cell[0] cell[8]
✍(◔◡◔) // Print 'o'

👈(⌒▽⌒)👉 -> 255
Ꮚ(ò_óˇ)Ↄ 👈(⌒▽⌒)👉 -> 0

// 'r': 114 = cell[2] + cell[3] + cell[4] + cell[7]
Ꮚ(ò_óˇ)Ↄ 👈(⌒▽⌒)👉 // NAND cell[0]
o(( >ω<))o o(( >ω<))o Ꮚ(ò_óˇ)Ↄ // copy cell[2]
o((>ω< ))o o((>ω< ))o 👈(⌒▽⌒)👉 // NAND cell[0] cell[2]
Ꮚ(ò_óˇ)Ↄ 👈(⌒▽⌒)👉
o(( >ω<))o o(( >ω<))o o(( >ω<))o Ꮚ(ò_óˇ)Ↄ // copy cell[3]
o((>ω< ))o o((>ω< ))o o((>ω< ))o 👈(⌒▽⌒)👉
Ꮚ(ò_óˇ)Ↄ 👈(⌒▽⌒)👉
o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o Ꮚ(ò_óˇ)Ↄ // copy cell[4]
o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o 👈(⌒▽⌒)👉
Ꮚ(ò_óˇ)Ↄ 👈(⌒▽⌒)👉
o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o((
>ω<))o Ꮚ(ò_óˇ)Ↄ // copy cell[7]
o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω<
))o 👈(⌒▽⌒)👉
✍(◔◡◔) // Print 'r'

👈(⌒▽⌒)👉 -> 255
Ꮚ(ò_óˇ)Ↄ 👈(⌒▽⌒)👉 -> 0

// 'l': 108 = cell[2] + cell[3] + cell[5] + cell[6]
Ꮚ(ò_óˇ)Ↄ 👈(⌒▽⌒)👉 // NAND cell[0]
o(( >ω<))o o(( >ω<))o Ꮚ(ò_óˇ)Ↄ // copy cell[2]
o((>ω< ))o o((>ω< ))o 👈(⌒▽⌒)👉 // NAND cell[0] cell[2]
Ꮚ(ò_óˇ)Ↄ 👈(⌒▽⌒)👉
o(( >ω<))o o(( >ω<))o o(( >ω<))o Ꮚ(ò_óˇ)Ↄ // copy cell[3]
o((>ω< ))o o((>ω< ))o o((>ω< ))o 👈(⌒▽⌒)👉
Ꮚ(ò_óˇ)Ↄ 👈(⌒▽⌒)👉
o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o Ꮚ(ò_óˇ)Ↄ // copy
cell[5]
o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o 👈(⌒▽⌒)👉
Ꮚ(ò_óˇ)Ↄ 👈(⌒▽⌒)👉
o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o Ꮚ(ò_óˇ)Ↄ
// copy cell[6]
o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o 👈(⌒▽⌒)
👉
```

```
✍(◔◡◔) // Print 'l'

👈(⌒▽⌒)👉  -> 255
ᕦ(ò_óˇ)ᕤ 👈(⌒▽⌒)👉  -> 0

// 'd': 100 = cell[2] + cell[3] + cell[6]
ᕦ(ò_óˇ)ᕤ 👈(⌒▽⌒)👉
o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ
o((>ω< ))o o((>ω< ))o 👈(⌒▽⌒)👉
ᕦ(ò_óˇ)ᕤ 👈(⌒▽⌒)👉
o(( >ω<))o o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ
o((>ω< ))o o((>ω< ))o o((>ω< ))o 👈(⌒▽⌒)👉
ᕦ(ò_óˇ)ᕤ 👈(⌒▽⌒)👉
o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ
o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o 👈(⌒▽⌒)👉
✍(◔◡◔) // Print 'd'

👈(⌒▽⌒)👉  -> 255
ᕦ(ò_óˇ)ᕤ 👈(⌒▽⌒)👉  -> 0

// '!': 33 = cell[6] + cell[8]
ᕦ(ò_óˇ)ᕤ 👈(⌒▽⌒)👉
o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ
o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o 👈(⌒▽⌒)👉
ᕦ(ò_óˇ)ᕤ 👈(⌒▽⌒)👉
o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ
o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o 👈(⌒▽⌒)👉
✍(◔◡◔) // Print '!'
👈(⌒▽⌒)👉  -> 255
ᕦ(ò_óˇ)ᕤ 👈(⌒▽⌒)👉  -> 0

// '\n': 10 = cell[5] + cell[7]
ᕦ(ò_óˇ)ᕤ 👈(⌒▽⌒)👉
o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ
o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o 👈(⌒▽⌒)👉
ᕦ(ò_óˇ)ᕤ 👈(⌒▽⌒)👉
o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o o(( >ω<))o ᕦ(ò_óˇ)ᕤ
o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o o((>ω< ))o 👈(⌒▽⌒)👉
✍(◔◡◔) // Print '\n'
👈(⌒▽⌒)👉  -> 255
ᕦ(ò_óˇ)ᕤ 👈(⌒▽⌒)👉  -> 0
```

External Resources

Brainf*ck: https://esolangs.org/wiki/Brainfuck Github Repo for KaoLang (👉ﾟヮﾟ)👉 : https://github.com/Vbeelearncode/KaoLang KaoLang Interpreter: Coming soon