

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
(∩·_·)∩TTTT	Begin loop if current memory cell is not zero
(°□°)°∩TTTT	End loop — jump back to matching loop start if current cell is not zero
𐄂(☺☺☺)	Output the character corresponding to the current memory cell
(0_0)	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
𐄂(ò_ó)𐄂	Copy current memory cell value into the register if register value is 0, else replace memory cell value with value in register + set register value to 0

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
<	o(>ω<)o	Move memory pointer left

Brainf*ck	KaoLang (𐀀𐀁𐀂)𐀃 Equivalent	Description
+	(see below)	Increment current memory cell by 1 using NAND and shifts
-	(see below)	Decrement current memory cell by 1 using NAND and shifts
.	𐀄(𐀅𐀆𐀇)	Output the character at current memory cell
,	(𐀈𐀉)	Read one byte into current memory cell
[(𐀊𐀋𐀌)𐀍𐀎𐀏	Start loop if current cell is not 0
]	(𐀐𐀑𐀒) 𐀓𐀔 𐀕𐀖	Jump to matching [if current cell is not 0

+ — Increment Implementation

```

cell[0] stores a (original number); cell[1] stores b (= 1);
o(( >w< ))o                               // Move to cell[1]
𐀀(𐀁𐀂)𐀃                                     // NAND with self → 255
𐀄(𐀅𐀆)𐀇                                     // Copy 255 to register
(𐀈𐀉𐀊)𐀋𐀌                                     // Shift right → 127
𐀍(𐀎𐀏𐀐)𐀑                                     // Shift left → 254
𐀒(𐀓𐀔)𐀕                                     // NAND(255, 254) = 1; cell[1] = 1; register reset

(𐀊𐀋𐀌)𐀍𐀎𐀏                                     // While b ≠ 0
o(( >w< ))o 𐀄(𐀅𐀆)𐀇                             // Copy a to register
o(( >w< ))o o(( >w< ))o 𐀄(𐀅𐀆)𐀇 // Paste a to cell[2]
o(( >w< ))o 𐀄(𐀅𐀆)𐀇                             // Copy b
o(( >w< ))o 𐀀(𐀁𐀂)𐀃                             // NAND a b; register reset
𐀄(𐀅𐀆)𐀇                                     // Copy NAND result to reg
𐀀(𐀁𐀂)𐀃                                     // NAND again to get a & b; register reset

o(( >w< ))o 𐀄(𐀅𐀆)𐀇                             // Copy b to reg
o(( >w< ))o o(( >w< ))o 𐀄(𐀅𐀆)𐀇 // Paste b to cell[3]
𐀄(𐀅𐀆)𐀇 𐀀(𐀁𐀂)𐀃                             // ~b in cell[3]; register reset

o(( >w< ))o o(( >w< ))o o(( >w< ))o 𐀄(𐀅𐀆)𐀇 // Copy a
o(( >w< ))o o(( >w< ))o o(( >w< ))o o(( >w< ))o 𐀄(𐀅𐀆)𐀇 // Paste a to cell[4]
𐀄(𐀅𐀆)𐀇 𐀀(𐀁𐀂)𐀃                             // ~a in cell[4]; register reset

o(( >w< ))o o(( >w< ))o o(( >w< ))o o(( >w< ))o 𐀄(𐀅𐀆)𐀇 // Back to a, copy a
o(( >w< ))o o(( >w< ))o o(( >w< ))o // To ~b
𐀀(𐀁𐀂)𐀃                                     // cell[3] = NAND a ~b; register reset

o(( >w< ))o o(( >w< ))o 𐀄(𐀅𐀆)𐀇 // Copy b
o(( >w< ))o o(( >w< ))o o(( >w< ))o // To ~a
𐀀(𐀁𐀂)𐀃                                     // cell[4] = NAND b ~a; register reset

𐀄(𐀅𐀆)𐀇 o(( >w< ))o 𐀀(𐀁𐀂)𐀃 // Final XOR: (a NAND ~b) NAND (b NAND ~a);

```

register reset

```
Ⓢ(Ⓢ_Ⓢ)Ⓢ Ⓢ((>ω< ))Ⓢ Ⓢ((>ω< ))Ⓢ Ⓢ((>ω< ))Ⓢ Ⓢ(Ⓢ_Ⓢ)Ⓢ // Paste XOR result back to cell a
```

```
Ⓢ(( >ω< ))Ⓢ Ⓢ(( >ω< ))Ⓢ Ⓢ(Ⓢ_Ⓢ)Ⓢ // Copy carry
Ⓢ((>ω< ))Ⓢ Ⓢ(Ⓢ_Ⓢ)Ⓢ // Move to b and paste carry
Ⓢ(Ⓢ_Ⓢ)Ⓢ // b = carry << 1
```

```
(Ⓢ_Ⓢ)Ⓢ Ⓢ_Ⓢ // End loop
Ⓢ((>ω< ))Ⓢ // return to original value that has been incremented
```

- — Decrement Implementation

cell[0] stores a (original number); cell[1] stores b (= 255);

```
Ⓢ(( >ω< ))Ⓢ // Move to cell[1]
Ⓢ(Ⓢ_Ⓢ)Ⓢ // NAND with self → 255
```

```
(Ⓢ_Ⓢ)Ⓢ // While b ≠ 0
Ⓢ((>ω< ))Ⓢ Ⓢ(Ⓢ_Ⓢ)Ⓢ // Copy a to register
Ⓢ(( >ω< ))Ⓢ Ⓢ(( >ω< ))Ⓢ Ⓢ(Ⓢ_Ⓢ)Ⓢ // Paste a to cell[2]
Ⓢ((>ω< ))Ⓢ Ⓢ(Ⓢ_Ⓢ)Ⓢ // Copy b
Ⓢ(( >ω< ))Ⓢ Ⓢ(Ⓢ_Ⓢ)Ⓢ // NAND a b; register reset
Ⓢ(Ⓢ_Ⓢ)Ⓢ // Copy NAND result to reg
Ⓢ(Ⓢ_Ⓢ)Ⓢ // NAND again to get a & b; register reset
```

```
Ⓢ((>ω< ))Ⓢ Ⓢ(Ⓢ_Ⓢ)Ⓢ // Copy b to reg
Ⓢ(( >ω< ))Ⓢ Ⓢ(( >ω< ))Ⓢ Ⓢ(Ⓢ_Ⓢ)Ⓢ // Paste b to cell[3]
Ⓢ(Ⓢ_Ⓢ)Ⓢ Ⓢ(Ⓢ_Ⓢ)Ⓢ // ~b in cell[3]; register reset
```

```
Ⓢ((>ω< ))Ⓢ Ⓢ((>ω< ))Ⓢ Ⓢ((>ω< ))Ⓢ Ⓢ(Ⓢ_Ⓢ)Ⓢ // Copy a
Ⓢ(( >ω< ))Ⓢ Ⓢ(( >ω< ))Ⓢ Ⓢ(( >ω< ))Ⓢ Ⓢ(( >ω< ))Ⓢ Ⓢ(Ⓢ_Ⓢ)Ⓢ // Paste a to cell[4]
Ⓢ(Ⓢ_Ⓢ)Ⓢ Ⓢ(Ⓢ_Ⓢ)Ⓢ // ~a in cell[4]; register reset
```

```
Ⓢ((>ω< ))Ⓢ Ⓢ((>ω< ))Ⓢ Ⓢ((>ω< ))Ⓢ Ⓢ((>ω< ))Ⓢ Ⓢ(Ⓢ_Ⓢ)Ⓢ // Back to a, copy a
Ⓢ(( >ω< ))Ⓢ Ⓢ(( >ω< ))Ⓢ Ⓢ(( >ω< ))Ⓢ // To ~b
Ⓢ(Ⓢ_Ⓢ)Ⓢ // cell[3] = NAND a ~b; register reset
```

```
Ⓢ((>ω< ))Ⓢ Ⓢ((>ω< ))Ⓢ Ⓢ(Ⓢ_Ⓢ)Ⓢ // Copy b
Ⓢ(( >ω< ))Ⓢ Ⓢ(( >ω< ))Ⓢ Ⓢ(( >ω< ))Ⓢ // To ~a
Ⓢ(Ⓢ_Ⓢ)Ⓢ // cell[4] = NAND b ~a; register reset
```

```
Ⓢ(Ⓢ_Ⓢ)Ⓢ Ⓢ((>ω< ))Ⓢ Ⓢ(Ⓢ_Ⓢ)Ⓢ // Final XOR: (a NAND ~b) NAND (b NAND ~a); register reset
```

```
Ⓢ(Ⓢ_Ⓢ)Ⓢ Ⓢ((>ω< ))Ⓢ Ⓢ((>ω< ))Ⓢ Ⓢ((>ω< ))Ⓢ Ⓢ(Ⓢ_Ⓢ)Ⓢ // Paste XOR result back to cell a
```

```
Ⓢ(( >ω< ))Ⓢ Ⓢ(( >ω< ))Ⓢ Ⓢ(Ⓢ_Ⓢ)Ⓢ // Copy carry
```

Example Programs

4 / 9

```

C(ð_ó~)D ƒ(∇)G o(>w< ))o // Copy value to register, NAND with self, move
to next cell
C(ð_ó~)D ƒ(∇)G o(>w< ))o // Copy value to register, NAND with self, move
to next cell
C(ð_ó~)D ƒ(∇)G o(>w< ))o // Copy value to register, NAND with self, move
to next cell
C(ð_ó~)D ƒ(∇)G o(>w< ))o // Copy value to register, NAND with self, move
to next cell
C(ð_ó~)D ƒ(∇)G // Copy value to register, NAND with self, move to next cell

o(>w< ))o // move back to cell[0]

'H': 72 = cell[2] + cell[5]
C(ð_ó~)D ƒ(∇)G // NAND cell[0]
o(( >w< ))o o(( >w< ))o C(ð_ó~)D // copy cell[2]
o(>w< ))o o(>w< ))o ƒ(∇)G // NAND cell[0] cell[2]

C(ð_ó~)D ƒ(∇)G // NAND cell[0]
o(( >w< ))o o(( >w< ))o o(( >w< ))o o(( >w< ))o o(( >w< ))o C(ð_ó~)D // copy cell[5]
o(>w< ))o o(>w< ))o o(>w< ))o o(>w< ))o o(>w< ))o ƒ(∇)G // NAND cell[0]
cell[5]
A(☉☉) // Print 'H'

reset cell[0]
ƒ(∇)G -> 255
C(ð_ó~)D ƒ(∇)G -> 0

'e': 101 = cell[2] + cell[3] + cell[6] + cell[8]
C(ð_ó~)D ƒ(∇)G // NAND cell[0]
o(( >w< ))o o(( >w< ))o C(ð_ó~)D // copy cell[2]
o(>w< ))o o(>w< ))o ƒ(∇)G // NAND cell[0] cell[2]

C(ð_ó~)D ƒ(∇)G // NAND cell[0]
o(( >w< ))o o(( >w< ))o o(( >w< ))o C(ð_ó~)D // copy cell[3]
o(>w< ))o o(>w< ))o o(>w< ))o ƒ(∇)G // NAND cell[0] cell[3]

C(ð_ó~)D ƒ(∇)G // NAND cell[0]
o(( >w< ))o o(( >w< ))o o(( >w< ))o o(( >w< ))o o(( >w< ))o o(( >w< ))o C(ð_ó~)D //
copy cell[6]
o(>w< ))o o(>w< ))o o(>w< ))o o(>w< ))o o(>w< ))o o(>w< ))o ƒ(∇)G //
NAND cell[0] cell[6]

C(ð_ó~)D ƒ(∇)G // NAND cell[0]
o(( >w< ))o o(( >w< ))o o(( >w< ))o o(( >w< ))o o(( >w< ))o o(( >w< ))o o(( >w< ))o o((
>w< ))o C(ð_ó~)D // copy cell[8]
o(>w< ))o o(>w< ))o o(>w< ))o o(>w< ))o o(>w< ))o o(>w< ))o o(>w< ))o o(>w< ))o
o(>w< ))o ƒ(∇)G // NAND cell[0] cell[8]

A(☉☉) // Print 'e'

reset cell[0]
ƒ(∇)G -> 255
C(ð_ó~)D ƒ(∇)G -> 0

```

```

// 'l': 108 = cell[2] + cell[3] + cell[5] + cell[6]
C(ð_ó~)D Ǝ(∇)Ǝ // NAND cell[0]
o(( >w<))o o(( >w<))o C(ð_ó~)D // copy cell[2]
o((>w< ))o o((>w< ))o Ǝ(∇)Ǝ // NAND cell[0] cell[2]

C(ð_ó~)D Ǝ(∇)Ǝ // NAND cell[0]
o(( >w<))o o(( >w<))o o(( >w<))o C(ð_ó~)D // copy cell[3]
o((>w< ))o o((>w< ))o o((>w< ))o Ǝ(∇)Ǝ // NAND cell[0] cell[3]

C(ð_ó~)D Ǝ(∇)Ǝ // NAND cell[0]
o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o C(ð_ó~)D // copy cell[5]
o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o Ǝ(∇)Ǝ // NAND cell[0]
cell[5]

C(ð_ó~)D Ǝ(∇)Ǝ // NAND cell[0]
o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o C(ð_ó~)D //
copy cell[6]
o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o Ǝ(∇)Ǝ //
NAND cell[0] cell[6]
A(☹☹) // Print 'l'
A(☹☹) // Print 'l' one more time

// 'o': 111 = cell[2] + cell[3] + cell[5] + cell[6] + cell[7] + cell[8]
C(ð_ó~)D Ǝ(∇)Ǝ // NAND cell[0]
o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o
C(ð_ó~)D // copy cell[7]
o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o Ǝ
(∇)Ǝ // NAND cell[0] cell[7]

C(ð_ó~)D Ǝ(∇)Ǝ // NAND cell[0]
o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o((
>w<))o C(ð_ó~)D // copy cell[8]
o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o
o((>w< ))o Ǝ(∇)Ǝ // NAND cell[0] cell[8]
A(☹☹) // Print 'o'

// ',': 44 = cell[3] + cell[5] + cell[6]
C(ð_ó~)D Ǝ(∇)Ǝ // NAND cell[0]
o(( >w<))o o(( >w<))o o(( >w<))o C(ð_ó~)D // copy cell[3]
o((>w< ))o o((>w< ))o o((>w< ))o Ǝ(∇)Ǝ // NAND cell[0] cell[3]
C(ð_ó~)D Ǝ(∇)Ǝ // NAND cell[0]
o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o C(ð_ó~)D // copy cell[5]
o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o Ǝ(∇)Ǝ // NAND cell[0]
cell[5]
C(ð_ó~)D Ǝ(∇)Ǝ // NAND cell[0]
o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o C(ð_ó~)D //
copy cell[6]
o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o Ǝ(∇)Ǝ //
NAND cell[0] cell[6]
A(☹☹) // Print ','
Ǝ(∇)Ǝ -> 255
C(ð_ó~)D Ǝ(∇)Ǝ -> 0

// ' ': 32 = cell[3]

```

```

C(ð_ó~)D E(∇)E // NAND cell[0]
o(( >w<))o o(( >w<))o o(( >w<))o C(ð_ó~)D // copy cell[3]
o((>w< ))o o((>w< ))o o((>w< ))o E(∇)E // NAND cell[0] cell[3]
A(☉☉) // Print ' '
E(∇)E -> 255
C(ð_ó~)D E(∇)E -> 0

// 'W': 87 = cell[2] + cell[4] + cell[6] + cell[7] + cell[8]
C(ð_ó~)D E(∇)E // NAND cell[0]
o(( >w<))o o(( >w<))o C(ð_ó~)D // copy cell[2]
o((>w< ))o o((>w< ))o E(∇)E // NAND cell[0] cell[2]
C(ð_ó~)D E(∇)E // NAND cell[0]
o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o C(ð_ó~)D // copy cell[4]
o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o E(∇)E // NAND cell[0] cell[4]
C(ð_ó~)D E(∇)E // NAND cell[0]
o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o C(ð_ó~)D //
copy cell[6]
o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o E(∇)E //
NAND cell[0] cell[6]
C(ð_ó~)D E(∇)E // NAND cell[0]
o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o
C(ð_ó~)D // copy cell[7]
o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o E
(∇)E // NAND cell[0] cell[7]
C(ð_ó~)D E(∇)E // NAND cell[0]
o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o((
>w<))o C(ð_ó~)D // copy cell[8]
o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o
o((>w< ))o E(∇)E // NAND cell[0] cell[8]
A(☉☉) // Print 'W'
E(∇)E -> 255
C(ð_ó~)D E(∇)E -> 0

// 'o': 111 = cell[2] + cell[3] + cell[5] + cell[6] + cell[7] + cell[8]
C(ð_ó~)D E(∇)E // NAND cell[0]
o(( >w<))o o(( >w<))o C(ð_ó~)D // copy cell[2]
o((>w< ))o o((>w< ))o E(∇)E // NAND cell[0] cell[2]
C(ð_ó~)D E(∇)E // NAND cell[0]
o(( >w<))o o(( >w<))o o(( >w<))o C(ð_ó~)D // copy cell[3]
o((>w< ))o o((>w< ))o o((>w< ))o E(∇)E // NAND cell[0] cell[3]
C(ð_ó~)D E(∇)E // NAND cell[0]
o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o C(ð_ó~)D // copy cell[5]
o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o E(∇)E // NAND cell[0]
cell[5]
C(ð_ó~)D E(∇)E // NAND cell[0]
o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o C(ð_ó~)D //
copy cell[6]
o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o E(∇)E //
NAND cell[0] cell[6]
C(ð_ó~)D E(∇)E // NAND cell[0]
o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o
C(ð_ó~)D // copy cell[7]
o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o E
(∇)E // NAND cell[0] cell[7]

```

```

C(ð_ó~)Þ ƒ(∇)ƒ // NAND cell[0]
o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o
C(ð_ó~)Þ // copy cell[8]
o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o
o((>w< ))o ƒ(∇)ƒ // NAND cell[0] cell[8]
A(☹☹) // Print 'o'

```

```

ƒ(∇)ƒ -> 255
C(ð_ó~)Þ ƒ(∇)ƒ -> 0

```

```

// 'r': 114 = cell[2] + cell[3] + cell[4] + cell[7]
C(ð_ó~)Þ ƒ(∇)ƒ // NAND cell[0]
o(( >w<))o o(( >w<))o C(ð_ó~)Þ // copy cell[2]
o((>w< ))o o((>w< ))o ƒ(∇)ƒ // NAND cell[0] cell[2]
C(ð_ó~)Þ ƒ(∇)ƒ
o(( >w<))o o(( >w<))o o(( >w<))o C(ð_ó~)Þ // copy cell[3]
o((>w< ))o o((>w< ))o o((>w< ))o ƒ(∇)ƒ
C(ð_ó~)Þ ƒ(∇)ƒ
o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o C(ð_ó~)Þ // copy cell[4]
o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o ƒ(∇)ƒ
C(ð_ó~)Þ ƒ(∇)ƒ
o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o
C(ð_ó~)Þ // copy cell[7]
o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o ƒ
(∇)ƒ
A(☹☹) // Print 'r'

```

```

ƒ(∇)ƒ -> 255
C(ð_ó~)Þ ƒ(∇)ƒ -> 0

```

```

// 'l': 108 = cell[2] + cell[3] + cell[5] + cell[6]
C(ð_ó~)Þ ƒ(∇)ƒ // NAND cell[0]
o(( >w<))o o(( >w<))o C(ð_ó~)Þ // copy cell[2]
o((>w< ))o o((>w< ))o ƒ(∇)ƒ // NAND cell[0] cell[2]
C(ð_ó~)Þ ƒ(∇)ƒ
o(( >w<))o o(( >w<))o o(( >w<))o C(ð_ó~)Þ // copy cell[3]
o((>w< ))o o((>w< ))o o((>w< ))o ƒ(∇)ƒ
C(ð_ó~)Þ ƒ(∇)ƒ
o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o C(ð_ó~)Þ // copy cell[5]
o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o ƒ(∇)ƒ
C(ð_ó~)Þ ƒ(∇)ƒ
o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o o(( >w<))o C(ð_ó~)Þ //
copy cell[6]
o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o ƒ(∇)ƒ
A(☹☹) // Print 'l'

```

```

ƒ(∇)ƒ -> 255
C(ð_ó~)Þ ƒ(∇)ƒ -> 0

```

```

// 'd': 100 = cell[2] + cell[3] + cell[6]
C(ð_ó~)Þ ƒ(∇)ƒ
o(( >w<))o o(( >w<))o C(ð_ó~)Þ
o((>w< ))o o((>w< ))o ƒ(∇)ƒ
C(ð_ó~)Þ ƒ(∇)ƒ

```



```

o(( >w< ))o o(( >w< ))o o(( >w< ))o Ǝ(ò_ó~)ᵀ
o((>w< ))o o((>w< ))o o((>w< ))o Ǝ(↖)↵
Ǝ(ò_ó~)ᵀ Ǝ(↖)↵
o(( >w< ))o o(( >w< ))o o(( >w< ))o o(( >w< ))o o(( >w< ))o o(( >w< ))o Ǝ(ò_ó~)ᵀ
o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o Ǝ(↖)↵
𐀀(☹☹) // Print 'd'

Ǝ(↖)↵ -> 255
Ǝ(ò_ó~)ᵀ Ǝ(↖)↵ -> 0

// '!': 33 = cell[6] + cell[8]
Ǝ(ò_ó~)ᵀ Ǝ(↖)↵
o(( >w< ))o o(( >w< ))o o(( >w< ))o o(( >w< ))o o(( >w< ))o o(( >w< ))o Ǝ(ò_ó~)ᵀ
o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o Ǝ(↖)↵
Ǝ(ò_ó~)ᵀ Ǝ(↖)↵
o(( >w< ))o o(( >w< ))o o(( >w< ))o o(( >w< ))o o(( >w< ))o o(( >w< ))o o(( >w< ))o o((
>w< ))o Ǝ(ò_ó~)ᵀ
o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o
o((>w< ))o Ǝ(↖)↵
𐀀(☹☹) // Print '!'
Ǝ(↖)↵ -> 255
Ǝ(ò_ó~)ᵀ Ǝ(↖)↵ -> 0

// '\n': 10 = cell[5] + cell[7]
Ǝ(ò_ó~)ᵀ Ǝ(↖)↵
o(( >w< ))o o(( >w< ))o o(( >w< ))o o(( >w< ))o o(( >w< ))o Ǝ(ò_ó~)ᵀ
o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o Ǝ(↖)↵
Ǝ(ò_ó~)ᵀ Ǝ(↖)↵
o(( >w< ))o o(( >w< ))o o(( >w< ))o o(( >w< ))o o(( >w< ))o o(( >w< ))o o(( >w< ))o
Ǝ(ò_ó~)ᵀ
o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o o((>w< ))o Ǝ
(↖)↵
𐀀(☹☹) // Print '\n'
Ǝ(↖)↵ -> 255
Ǝ(ò_ó~)ᵀ Ǝ(↖)↵ -> 0

```