a 01 01 b 02 02 c 03 03 d 04 04 e 05 05 f 06 06 g 07 07 h 08 08 i 09 09

j 10 01 k 11 02 l 12 03 m 13 04 n 14 05 o 15 06 p 16 07 q 17 08 r 18 09

s 19 01 t 20 02 u 21 03 v 22 04 w 23 05 x 24 06 y 25 07 z 26 08

1 a j s 2 b k t (11 -> 2) 3 c l u 4 d m v (22 -> 4) 5 e n w 6 f o x (33 -> 6) 7 g p y 8 h q z 9 i r

https://en.wikipedia.org/wiki/Luoshu_Square

Here is a simple algorithm in Rust that converts a letter from the first set to the second set:

```
fn convert_letter(c: char) -> char { let offset = (c as u8 - 'a' as u8) * 2 + 1; (offset as char).to_uppercase().next().unwrap() }
```

And here is the algorithm to convert a letter from the second set back to the first set:

```
fn convert_letter_back(c: char) -> char { let offset = (c as u8 - 'A' as u8) / 2; (offset as char + 'a' as u8) as char } assert_eq!(convert_to_number("abc"), 123); assert_eq!(convert_to_number("jst"), 578); assert_eq!(convert_to_letter(123), "abc"); assert_eq!(convert_to_letter(578), "jst"); fn convert_to_number(s: &str) -> u32 { let mut result = 0; for c in s.chars() { let offset = (c as u32) - ('a' as u32); result = result * 10 + (offset / 3) + 1; } result } fn convert_to_letter(n: u32) -> String { let mut result = String::new(); let mut n = n; while n > 0 { let digit = (n % 10) as u8 - 1; n /= 10; result.push((digit * 3)
```

```
+  'a'  as  u8)  as  char);     }     result.chars
().rev().collect()}
```