Digit 1

	W (1 - 9)	X	Y	Z (=> 0)
inp w				
mul x 0		0		
add x z		Z		
mod x 26		X % 26		
div z 1				
add x 10		(x % 26) + 10		
eql x w		0		
eql x 0		1		
mul y 0			0	
add y 25			25	
mul y x			25	
add y 1			26	
mul z y				26 * Z
mul y 0			0	
add y w			W	
add y 15			W + 15	
mul y x			W +15	
add z y				W + 15 + 26Z

Digit 2

	W (1 - 9)	X	Y	Z (=> 0)
inp w				
mul x 0		0		
add x z		Z		
mod x 26		X % 26		
div z 1				
add x 12		(x % 26) + 12		
eql x w		0		
eql x 0		1		
mul y 0			0	
add y 25			25	
mul y x			25	
add y 1			26	
mul z y				26 * Z
mul y 0			0	
add y w			W	
add y 8			W + 8	
mul y x			W + 8	
add z y				W + 8 + 26Z

Digit 3

	W (1 - 9)	X	Y	Z (=> 0)
inp w				
mul x 0		0		
add x z		Z		
mod x 26		X % 26		
div z 1				
add x 15		(x % 26) + 15		
eql x w		0		
eql x 0		1		
mul y 0			0	
add y 25			25	
mul y x			25	
add y 1			26	
mul z y				26 * Z
mul y 0			0	
add y w			W	
add y 2			W + 2	
mul y x			W + 2	
add z y				W + 2 + 26Z
inp w				

Digit 4

	W (1 - 9)	X	Y	Z (=> 0)
inp w				
mul x 0		0		
add x z		Z		
mod x 26		Z % 26		
div z 26		Z % 26		Fl(Z / 26)
add x -9		(Z % 26) - 9		
eql x w		0 of 1		
eql x 0		1 of 0		
mul y 0			0	
add y 25			25	
mul y x			25 of 0	
add y 1			26 of 1	
mul z y				26 * Fl(Z / 26) of Fl(Z / 26)
mul y 0			0	
add y w			W	
add y 6			W + 6	
mul y x			W + 6 of 0	
add z y				W + 6 + 26 * Fl(Z / 26) OF Fl(Z / 26)

Known: Z must be 0 for a set of input-digits to be valid
Assumption: 7 steps multiply the value by roughly a factor of 26
-> 7 equal steps must reduce to ever reach zero.
-> W + 6 + 26 * Fl(Z / 26) OF Fl(Z / 26)
-> W + 6 + 26 * (Z / 26) does not reduce Z, but roughly keeps it the same
-> Z / 26 does reduce Z by a factor of about 26
-> eql x w must be true, so since x = (Z % 26) - <some factor>, it must be equal to the input digit
-> given Z from previous calculations, you can deterministically find the input digit (maybe use this to check if the digit is in a valid input space to reduce calculations?

DIGIT	FORMULA
1	W + 15 + 26Z
2	W + 8 + 26Z
3	W + 2 + 26Z
4	W = (Z % 26) - 9
5	W + 13 +26Z
6	W + 4 + 26Z
7	W + 1 + 26Z
8	W = (Z % 26) - 5
9	W + 5 + 26Z
10	W = (Z % 26) - 7
11	W = (Z % 26) - 12
12	W = (Z % 26) - 10
13	W = (Z % 26) - 1
14	W = (Z % 26) - 11

PSEUDO CODE:

```
Pub fn main() -> i64 {
    For x in 1111111..9999999
        If x contains 0 {
            Continue
        }
        Let res = solve(digits)
        If Some(res) {
            Return digits
        }
    }
}
```

```
Weights_nondeter = [15, 8, 2, 13, 4, 1, 5]
Weights_deter = [9, 5, 7, 12, 10, 1, 11]
Fn solve(digits) -> Optional<i64> {
    Let z = 0
    Let res = []
    Let digits_idx = 0
    For i in 0..14 {
        If deter {
            Let digit = ((z % 26) - weights_deter[idx])
            If digit < 1 || digit > 9  {
                Return None
            }
            Z = floor(z / 26)
            Res.push(digit)
        } else {
            Z = digits[digits_idx] + weights_nondeter[idx] + z * 26
            Res.push(digits[digits_idx++])
        }
    }

    Return res
}
```