Advent of Code 2025 in Typst 😀

Contents

day 1

Problem summary

Given a **sequence** of dial **rotations** on a **circular scale from 0 to 99**, **starting at** position **50**, determine **how often** the dial **reaches position 0** after applying each rotation in order.

Mathematical formulation

- Let the dial positions be elements of the cyclic group $\mathbb{Z}_{100}$.
- Let the initial position be $x_0 = 50$.
- Let the sequence of n rotations be (d_i, k_i) with $d_i \in \{L, R\}$ and $k_i \in \mathbb{N}$.

Define the signed rotation

$$s_i = \begin{cases} -k_i \text{ if } d_i = L \\ k_i \text{ if } d_i = R \end{cases}$$

and the position update

$$x_i \equiv x_{i-1} + s_i \pmod{100}, \quad i = 1, ..., n.$$

The password is the number of indices i for which the dial reaches zero

$$P = |\{i \in \{1, ..., n\} : x_i = 0\}|.$$

Mathematische Formulierung

Die Drehscheibe besitzt die Positionen des zyklischen Raums $\mathbb{Z}_{100}$. Der Anfangswert ist $x_0 = 50$.

Die Eingabe sei eine Folge von ganzen Zahlen $r_1, r_2, ..., r_n$, wobei jede Zahl bereits ein Vorzeichen trägt. Ein Eintrag ist negativ, falls die ursprüngliche Richtung L war, und positiv, falls die ursprüngliche Richtung R war.

Die Aktualisierung der Position erfolgt durch

$$x_i \equiv x_{i-1} + r_i \pmod{100} \quad \text{für } i = 1, ..., n.$$

Das Passwort ergibt sich aus der Anzahl der Schritte, in denen die Position den Wert null annimmt:

$$P = |\{i \in \{1, ..., n\} : x_i = 0\}|.$$

Python3

```python
>>> from main import xs
>>>
>>> xs
'\nL68\nL30\nR48\nL5\nR60\nL55\nL1\nL99\nR14\nL82\n'
>>>
>>> xs.split()
['L68', 'L30', 'R48', 'L5', 'R60', 'L55', 'L1', 'L99', 'R14', 'L82']
>>>
>>> xs.replace("L", "+")
'\n+68\n+30\nR48\n+5\nR60\n+55\n+1\n+99\nR14\n+82\n'
>>>
>>> xs.replace("L", "+").replace("R", "-")
'\n+68\n+30\n-48\n+5\n-60\n+55\n+1\n+99\n-14\n+82\n'
>>>
>>> xs.replace("L", "+").replace("R", "-").split()
['+68', '+30', '-48', '+5', '-60', '+55', '+1', '+99', '-14', '+82']
>>>
>>> [int(e) for e in xs.replace("L", "+").replace("R", "-").split()]
[68, 30, -48, 5, -60, 55, 1, 99, -14, 82]
>>>
>>> list( map(int, xs.replace("L", "+").replace("R", "-").split())  )
[68, 30, -48, 5, -60, 55, 1, 99, -14, 82]
>>>
>>> (50 + 68) % 100
18
>>>
>>> (50 - 68) % 100
82
>>>
>>> list( map(int, xs.replace("L", "-").replace("R", "+").split())  )
[-68, -30, 48, -5, 60, -55, -1, -99, 14, -82]
```

⇝ Define the signed rotation

$$s_i = \begin{cases} -k_i & \text{if } d_i = L \\ k_i & \text{if } d_i = R \end{cases}$$

```python
>>>
>>> for r in map(int, xs.replace("L", "-").replace("R", "+").split()):
...     temp = (temp + r) % 100
```

⇝ and the position update

$$x_i \equiv x_{i-1} + s_i \pmod{100}, \quad i = 1, ..., n.$$

```python
...     if temp == 0: count += 1
```

⇝ The password is the number of indices i for which the dial reaches zero

$$P = |\{i \in \{1, ..., n\} : x_i = 0\}|.$$

```python
...
>>> print(count)
3
>>>
>>> (lambda xs: (
...     lambda pos, c:
...         ([(pos := (pos + r) % 100,
...             c := c + (pos == 0))
...            for r in map(int, xs.replace("L", "-").replace("R", "+").split())],
...           c)[1]
...  )(50, 0))(xs)
3
```

typst

typst

```typst
#let count-zeros(xs) = {
  let steps = xs.replace("L", "-").replace("R", "+").split().map(int)
  let result = steps.fold((50, 0), (acc, r) => {
    let pos = calc.rem((acc.at(0) + r), 100)
    let cnt = acc.at(1) + int(pos == 0)
    (pos, cnt)
  })
  result.at(1)
}
```

typst

```typst
#count-zeros("L68 L30 R48 L5 R60 L55 L1 L99 R14 L82")
```

3

count-zeros

This function computes the AoC day 1 challenge, $\mathrm{sinc}(x) = \frac{\sin(x)}{x}$.

```typst
#count-zeros("L68 L30 R48
L5 R60 L55 L1 L99 R14
L82")
```

3

Parameters

```
count-zeros(xs: string) -> int
```

xs `string`

The argument