# Advent of Code



## Answers

### Part 1

Executed on the example input.
- Expected answer: **3**
- Computed answer: **3**

This seems correct

---

Executed on the evaluation input.
- Computed answer: **1147**

### Part 2

Executed on the example input.
- Expected answer: **6**
- Computed answer: **6**

This seems correct

---

Executed on the evaluation input.
- Computed answer: **6789**

# Logs

## Parsing

### Part 1

```
(-68, -30, 48, -5, 60, -55, -1, -99, 14, -82)
```

### Part 2

```
(Same)
```

## Part 1

```
```

## Part 2

```
> Move: -68
Went below and past 0 to -18
Pointing at 82
> Move: -30
Pointing at 52
> Move: 48
Full positive turn to 100
Pointing at 0
> Move: -5
Pointing at 95
> Move: 60
Full positive turn to 155
Pointing at 55
> Move: -55
Went below and past 0 to 0
Pointing at 0
> Move: -1
Pointing at 99
> Move: -99
Went below and past 0 to 0
Pointing at 0
> Move: 14
Pointing at 14
> Move: -82
```

```
Went below and past 0 to -68
Pointing at 32
```

# Source code

## Preliminaries

```
#import "/template/aot.typ"

#show: aot.format

#aot.parser(input => {
  input.split("\n").filter(s => s != "").map(instr => {
    let dir = instr.at(0)
    let num = int(instr.slice(1))
    if dir == "L" { -num } else { num }
  })
})
```

## Part 1

```
#aot.solve(input => {
  let cur = 50
  let zeroed = 0
  for mv in input {
    cur = calc.rem(cur + mv, 100)
    if cur == 0 {
      zeroed += 1
    }
  }
  aot.answer(zeroed)
})
```

## Part 2

```
#aot.solve(input => {
  let cur = 50
  let zeroed = 0
  for mv in input {
    while mv > 100 {
      zeroed += 1
      mv -= 100
    }
    while mv < -100 {
      zeroed += 1
      mv += 100
    }
    let next = cur + mv
    if next <= 0 and cur > 0 {
```

```
      zeroed += 1
    } else if next == -100 and cur == 0 {
      zeroed += 1
    } else if next >= 100 {
      zeroed += 1
    }
    cur = calc.rem-euclid(next, 100)
  }
  aot.answer(zeroed)
})
```