Leetcode.typ

Gabriel Wu and contributors

December 30, 2025

Contents

0001. Two Sum

Given an array of integers `nums` and an integer `target`, return indices of the two numbers such that they add up to `target`.

You may assume that each input would have **exactly one solution**, and you may not use the same element twice.

You can return the answer in any order.

Test Results

Case 1

nums: [2, 7, 11, 15]
target: 9

Expected	Your Output
[0, 1]	none

Case 2

nums: [3, 2, 4]
target: 6

Expected	Your Output
[1, 2]	none

Case 3

nums: [3, 3]
target: 6

Expected	Your Output
[0, 1]	none

Case 4

nums: [0, 0]
target: 1

Expected	Your Output
[-1, -1]	none

Case 5

nums: [1, 4, 7, 10, 13, 16, 19, 22, 25, 28, 31, 34, 37, 40, 43, 46, 49, 52, 55, 58, 61, 64, 67, 70, 73, 76, 79, 82, 85, 88, 91, 94, 97]
target: 191

Expected	Your Output
[31, 32]	none

Reference Solution

```typst
#let solution-ref(arr, target) = {
  let d = (:)
  let ans = (-1, -1)
  for (i, num) in arr.enumerate() {
    if str(target - num) in d {
      ans = (d.at(str(target - num)), i)
      break
    } else {
      d.insert(str(num), i)
    }
  }

  ans
}
```

0002. Add Two Numbers

You are given two **non-empty** linked lists representing two non-negative integers. The digits are stored in **reverse order**, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

Test Results

Case 1

l1: 2→4→3
l2: 5→6→4

Expected	Your Output
7→0→8	none

Case 2

l1: 0
l2: 0

Expected	Your Output
0	none

Case 3

l1: 9→9→9→9→9→9→9
l2: 9→9→9→9

Expected	Your Output
8→9→9→9→0→0→0→1	none

Case 4

l1: 2→4→3
l2: 5→6→4→9

Expected	Your Output
7→0→8→9	none

Reference Solution

```typst
#let solution-ref(l1, l2) = {
  let p = ()
  let carry = 0
  while l1.val != none or l2.val != none {
    let x = if l1.val != none { l1.val } else { 0 }
    let y = if l2.val != none { l2.val } else { 0 }
    let sum = x + y + carry
    carry = calc.floor(sum / 10)
    p.push(calc.rem(sum, 10))
    if l1.next != none {
      l1 = l1.next
    }
    if l2.next != none {
      l2 = l2.next
    }
  }
  if carry > 0 {
    p.push(carry)
  }
  linkedlist(p)
}
```

0003. Longest Substring Without Repeating Characters

Given a string `s`, find the length of the **longest substring** without repeating characters.

Test Results

Case 1

s: "abcabcbb"

Expected	Your Output
3	none

Case 2

s: "bbbbb"

Expected	Your Output
1	none

Case 3

s: "pwwkew"

Expected	Your Output
3	none

Reference Solution

```typst
#let solution-ref(s) = {
  let s = s.clusters()
  let n = s.len()
  let ans = 1
  let l = 0
  let d = (:)
  for r in range(n) {
    let c = s.at(r)
    while c in d {
      let cl = s.at(l)
      let _ = d.remove(cl)
      l += 1
    }
```

```
14      d.insert(c, 1)
15      ans = calc.max(ans, r - l + 1)
16    }
17    ans
18  }
```

0004. Median of Two Sorted Arrays

Given two sorted arrays `nums1` and `nums2` of size `m` and `n` respectively, return the **median** of the two sorted arrays.

The overall run time complexity should be $\mathcal{O}(\log(m+n))$.

Test Results

Case 1

nums1: [1, 3]
nums2: [2]

Expected	Your Output
2	none

Case 2

nums1: [1, 2]
nums2: [3, 4]

Expected	Your Output
2.5	none

Case 3

nums1: [0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 51, 54, 57, 60, 63, 66, 69, 72, 75, 78, 81, 84, 87, 90, 93, 96, 99]
nums2: [0, 6, 12, 18, 24, 30, 36, 42, 48, 54, 60, 66, 72, 78, 84, 90, 96, 102, 108, 114, 120, 126, 132, 138, 144, 150, 156, 162, 168, 174, 180, 186, 192, 198]

Expected	Your Output
66.0	none

Reference Solution

```typst
#let solution-ref(nums1, nums2) = {
  if nums1.len() > nums2.len() {
    return solution-ref(nums2, nums1)
  }

  let imax = 4000000000
```

```
7     let imin = -4000000000
8
9     let m = nums1.len()
10    let n = nums2.len()
11    let left = 0
12    let right = m
13    let med1 = 0
14    let med2 = 0
15
16    while left <= right {
17      let i = calc.floor((left + right) / 2)
18      let j = calc.floor((m + n + 1) / 2) - i
19
20      let nums_im1 = if i == 0 { imin } else { nums1.at(i - 1) }
21      let nums_i = if i == m { imax } else { nums1.at(i) }
22      let nums_jm1 = if j == 0 { imin } else { nums2.at(j - 1) }
23      let nums_j = if j == n { imax } else { nums2.at(j) }
24
25      if nums_im1 <= nums_j {
26        med1 = calc.max(nums_im1, nums_jm1)
27        med2 = calc.min(nums_i, nums_j)
28        left = i + 1
29      } else {
30        right = i - 1
31      }
32    }
33
34    if calc.rem(m + n, 2) == 0 {
35      (med1 + med2) / 2
36    } else {
37      med1
38    }
39  }
```

0005. Longest Palindromic Substring

Given a string `s`, return the **longest palindromic substring** in `s`.

Test Results

Case 1

s: "babad"

Expected	Your Output
"bab"	none

Case 2

s: "cbbd"

Expected	Your Output
"bb"	none

Case 3

s: "abcdefgfedcbb"

Expected	Your Output
"bcdefgfedcb"	none

Case 4

s: "accc"

Expected	Your Output
"ccc"	none

Case 5

s: "a"

Expected	Your Output
"a"	none

Case 6

s: "aa"

Expected	Your Output
"aa"	none

Case 7

s: "asasfsafdaasfsaasa"

Expected	Your Output
"aasfsaa"	none

Reference Solution

```typst
#let solution-ref(s) = {
  let s = s.clusters()
  let t = ()
  for c in s {
    t.push("$")
    t.push(c)
  }
  t.push("$")
  let n = t.len()
  let a = fill(0, n)
  let l = 0
  let r = -1
  for i in range(n) {
    let j = if i > r { 1 } else { calc.min(a.at(l + r - i), r - i + 1) }
    while i >= j and i + j < n and t.at(i - j) == t.at(i + j) {
      j = j + 1
    }
    a.at(i) = j
    j = j - 1
    if i + j > r {
      l = i - j
      r = i + j
    }
  }

  let ans = 0
  let hi = 0
  for i in range(n) {
```

```
29      if a.at(i) > hi {
30        ans = i
31        hi = a.at(i)
32      }
33    }
34
35    let ret = ()
36    for i in range(ans - hi + 1, ans + hi) {
37      if t.at(i) != "$" {
38        ret.push(t.at(i))
39      }
40    }
41
42    ret.join()
43  }
```

0006. Zigzag Conversion

The string "PAYPALISHIRING" is written in a zigzag pattern on a given number of rows like this:

```
1 P   A   H   N
2 A P L S I I G
3 Y   I   R
```

And then read line by line: "PAHNAPLSIIGYIR"

Write the code that will take a string and make this conversion given a number of rows.

Test Results

Case 1

s: "PAYPALISHIRING"
numRows: 3

Expected	Your Output
"PAHNAPLSIIGYIR"	none

Case 2

s: "PAYPALISHIRING"
numRows: 4

Expected	Your Output
"PINALSIGYAHRPI"	none

Case 3

s: "A"
numRows: 1

Expected	Your Output
"A"	none

Reference Solution

```typst
#let solution-ref(s, numRows) = {
  if numRows == 1 {
    return s
  }
```

```
5
6     let s = s.clusters()
7     let n = s.len()
8     let ret = ()
9     let cycleLen = 2 * numRows - 2
10
11    for i in range(numRows) {
12      for j in range(0, n - i, step: cycleLen) {
13        ret.push(s.at(j + i))
14        if i != 0 and i != numRows - 1 and j + cycleLen - i < n {
15          ret.push(s.at(j + cycleLen - i))
16        }
17      }
18    }
19
20    ret.join()
21  }
```

0007. Reverse Integer

Given a signed 32-bit integer x, return x with its digits reversed. If reversing x causes the value to go outside the signed 32-bit integer range $[-2^{31}, 2^{31} - 1]$, then return 0.

Assume the environment does not allow you to store 64-bit integers (signed or unsigned).

Test Results

Case 1

x: 123

Expected	Your Output
321	none

Case 2

x: -123

Expected	Your Output
-321	none

Case 3

x: 120

Expected	Your Output
21	none

Case 4

x: 0

Expected	Your Output
0	none

Case 5

x: 23498423

Expected	Your Output

32489432	none

Case 6

x: -213898800

Expected	Your Output
-8898312	none

Case 7

x: 1534236469

Expected	Your Output
0	none

Case 8

x: 2147483647

Expected	Your Output
0	none

Case 9

x: -2147483648

Expected	Your Output
0	none

Reference Solution

```typst
#let solution-ref(x) = {
  let ans = 0
  let sign = if x >= 0 { 0 } else { -1 }
  if x == -2147483648 {
    return 0
  }

  x = calc.abs(x)
  while x != 0 {
```

```
10      let pop = calc.rem(x, 10)
11      x = calc.floor(x / 10)
12      if ans > 214748364 or (ans == 214748364 and pop - sign > 7) {
13        return 0
14      }
15      ans = ans * 10 + pop
16    }
17    ans * (2 * sign + 1)
18  }
```

0008. String to Integer (atoi)

Implement the `myAtoi(string s)` function, which converts a string to a 32-bit signed integer (similar to C/C++'s atoi function).

The algorithm for `myAtoi(string s)` is as follows:

1. Read in and ignore any leading whitespace.
2. Check if the next character (if not already at the end of the string) is '-' or '+'. Read this character in if it is either. This determines if the final result is negative or positive respectively. Assume the result is positive if neither is present.
3. Read in next the characters until the next non-digit character or the end of the input is reached. The rest of the string is ignored.
4. Convert these digits into an integer (i.e. "123" -> 123, "0032" -> 32). If no digits were read, then the integer is 0. Change the sign as necessary (from step 2).
5. If the integer is out of the 32-bit signed integer range $[-2^{31}, 2^{31} - 1]$, then clamp the integer so that it remains in the range. Specifically, integers less than -2^{31} should be clamped to -2^{31}, and integers greater than $2^{31} - 1$ should be clamped to $2^{31} - 1$.
6. Return the integer as the final result.

Note:

- Only the space character ' ' is considered a whitespace character.
- **Do not ignore** any characters other than the leading whitespace or the rest of the string after the digits.

Test Results

Case 1

s: "42"

Expected	Your Output
42	none

Case 2

s: " -42"

Expected	Your Output
-42	none

Case 3

s: "4193 with words"

Expected	Your Output
4193	none

Reference Solution

```typst
#let solution-ref(s) = {
  let numerics = "0123456789"
  let d = (
    "0": 0,
    "1": 1,
    "2": 2,
    "3": 3,
    "4": 4,
    "5": 5,
    "6": 6,
    "7": 7,
    "8": 8,
    "9": 9,
  )
  let s = s.clusters()
  let n = s.len()
  let i = 0
  let sign = 1
  let ans = 0
  while i < n and s.at(i) == " " {
    i += 1
  }
  if i < n and s.at(i) == "-" {
    sign = -1
    i += 1
  } else if i < n and s.at(i) == "+" {
    i += 1
  }
  while i < n and s.at(i) in numerics {
    if ans > 214748364 or (ans == 214748364 and d.at(s.at(i)) > 7) {
      if sign == 1 {
        return 2147483647
      } else {
        return -2147483648
      }
    }
    ans = ans * 10 + d.at(s.at(i))
    i += 1
  }
```

```
40    ans * sign
41  }
```

0009. Palindrome Number

Given an integer `x`, return `true` if `x` is a **palindrome**, and `false` otherwise.

Test Results

Case 1

x: 121

Expected	Your Output
true	none

Case 2

x: -121

Expected	Your Output
false	none

Case 3

x: 10

Expected	Your Output
false	none

Reference Solution

typst

```
#let solution-ref(x) = {
  if x < 0 {
    return false
  }
  if x == 0 {
    return true
  }
  if calc.rem(x, 10) == 0 {
    return false
  }

  let rev = 0
  while x > rev {
```

```
14      if rev > 214748364 or (rev == 214748364 and calc.rem(x, 10) > 7) {
15        return false
16      }
17      rev = rev * 10 + calc.rem(x, 10)
18      x = calc.floor(x / 10)
19    }
20    x == rev or x == calc.floor(rev / 10)
21  }
```

0010. Regular Expression Matching

Given an input string s and a pattern p, implement regular expression matching with support for `'.'` and `'*'` where:

- `'.'` Matches any single character.
- `'*'` Matches zero or more of the preceding element.

The matching should cover the **entire** input string (not partial).

Test Results

Case 1

s: "aa"
p: "a"

Expected	Your Output
false	none

Case 2

s: "aa"
p: "a*"

Expected	Your Output
true	none

Case 3

s: "ab"
p: ".*"

Expected	Your Output
true	none

Case 4

s: "aab"
p: "c*a*b"

Expected	Your Output
true	none

Case 5

s: "mississippi"
p: "mis*is*p*."

Expected	Your Output
false	none

Case 6

s: "ab"
p: ".*c"

Expected	Your Output
false	none

Case 7

s: "ab"
p: ".*c*"

Expected	Your Output
true	none

Case 8

s: "香蕉 x 牛奶"
p: "香.*牛."

Expected	Your Output
true	none

Reference Solution

```typst
#let solution-ref(s, p) = {
  let s = s.clusters()
  let p = p.clusters()
  let m = s.len()
  let n = p.len()
  let dp = fill(fill(false, n + 1), m + 1)
  dp.at(0).at(0) = true
  for i in range(0, m + 1) {
    for j in range(1, n + 1) {
```

```
      if p.at(j - 1) != "*" {
        if i > 0 and (p.at(j - 1) == "." or p.at(j - 1) == s.at(i - 1)) {
          dp.at(i).at(j) = dp.at(i - 1).at(j - 1)
        }
      } else {
        if j >= 2 {
          dp.at(i).at(j) = dp.at(i).at(j) or dp.at(i).at(j - 2)
        }
        if (
          i >= 1
            and j >= 2
            and (p.at(j - 2) == "." or p.at(j - 2) == s.at(i - 1))
        ) {
          dp.at(i).at(j) = dp.at(i).at(j) or dp.at(i - 1).at(j)
        }
      }
    }
  }
  dp.at(m).at(n)
}
```

0011. Container With Most Water

You are given an integer array height of length n. There are n vertical lines drawn such that the two endpoints of the i^{th} line are `(i, 0)` and `(i, height[i])`.

Find two lines that together with the x-axis form a container, such that the container contains the most water.

Return the maximum amount of water a container can store.

Notice that you may not slant the container.

Test Results

Case 1

height: [1, 8, 6, 2, 5, 4, 8, 3, 7]

Expected	Your Output
49	none

Case 2

height: [1, 1]

Expected	Your Output
1	none

Reference Solution

```typst
#let solution-ref(height) = {
  let n = height.len()
  let l = 0
  let r = n - 1
  let ans = 0
  while l < r {
    ans = calc.max(ans, calc.min(height.at(l), height.at(r)) * (r - l))
    if height.at(l) < height.at(r) {
      l += 1
    } else {
      r -= 1
    }
  }
  ans
}
```

0012. Integer to Roman

Roman numerals are represented by seven different symbols: `I`, `V`, `X`, `L`, `C`, `D` and `M`.

Symbol	Value
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

For example, `2` is written as `II` in Roman numeral, just two one's added together. `12` is written as `XII`, which is simply `X + II`. The number `27` is written as `XXVII`, which is `XX + V + II`.

Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not `IIII`. Instead, the number four is written as `IV`. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as `IX`. There are six instances where subtraction is used:

- `I` can be placed before `V` (5) and `X` (10) to make 4 and 9.
- `X` can be placed before `L` (50) and `C` (100) to make 40 and 90.
- `C` can be placed before `D` (500) and `M` (1000) to make 400 and 900.

Given an integer, convert it to a roman numeral.

Test Results

Case 1

num: 3

Expected	Your Output
"III"	none

Case 2

num: 58

Expected	Your Output
"LVIII"	none

Case 3

num: 1994

Expected	Your Output
"MCMXCIV"	none

Reference Solution

```typst
#let solution-ref(num) = {
  numbering("I", num)
}
```

0013. Roman to Integer

Roman numerals are represented by seven different symbols: `I`, `V`, `X`, `L`, `C`, `D` and `M`.

Symbol	Value
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

For example, `2` is written as `II` in Roman numeral, just two one's added together. `12` is written as `XII`, which is simply `X + II`. The number `27` is written as `XXVII`, which is `XX + V + II`.

Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not `IIII`. Instead, the number four is written as `IV`. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as `IX`. There are six instances where subtraction is used:

- `I` can be placed before `V` (5) and `X` (10) to make 4 and 9.
- `X` can be placed before `L` (50) and `C` (100) to make 40 and 90.
- `C` can be placed before `D` (500) and `M` (1000) to make 400 and 900.

Given a roman numeral, convert it to an integer.

Test Results

Case 1

s: "III"

Expected	Your Output
3	none

Case 2

s: "LVIII"

Expected	Your Output
58	none

Case 3

s: "MCMXCIV"

Expected	Your Output
1994	none

Reference Solution

```typst
#let solution-ref(s) = {
  let s = s.clusters()
  let n = s.len()
  let d = ("I": 1, "V": 5, "X": 10, "L": 50, "C": 100, "D": 500, "M": 1000)
  let ans = 0
  for i in range(n) {
    if i < n - 1 and d.at(s.at(i)) < d.at(s.at(i + 1)) {
      ans -= d.at(s.at(i))
    } else {
      ans += d.at(s.at(i))
    }
  }
  ans
}
```

0014. Longest Common Prefix

Write a function to find the longest common prefix string amongst an array of strings.

If there is no common prefix, return an empty string `""`.

Test Results

Case 1

strs: ["flower", "flow", "flight"]

Expected	Your Output
"fl"	none

Case 2

strs: ["dog", "racecar", "car"]

Expected	Your Output
""	none

Reference Solution

```typst
#let solution-ref(strs) = {
  let strs = strs.map(x => x.clusters())
  let i = 0
  let n = strs.len()
  let ans = ("",) // We put an empty string here to avoid returning none
  while i >= 0 {
    for j in range(n) {
      if strs.at(j).len() <= i or strs.at(j).at(i) != strs.at(0).at(i) {
        i = -1
        break
      }
    }
    if i != -1 {
      ans.push(strs.at(0).at(i))
      i += 1
    }
  }
  ans.join()
}
```

0015. 3Sum

Given an integer array nums, return all the triplets `[nums[i], nums[j], nums[k]]` such that `i != j`, `i != k`, and `j != k`, and `nums[i] + nums[j] + nums[k] == 0`.

Notice that the solution set must not contain duplicate triplets.

Test Results

Case 1

nums: [-1, 0, 1, 2, -1, -4]

Expected	Your Output
[[-1, -1, 2], [-1, 0, 1]]	none

Case 2

nums: [0, 1, 1]

Expected	Your Output
[]	none

Case 3

nums: [0, 0, 0]

Expected	Your Output
[[0, 0, 0]]	none

Case 4

nums: [-10, -7, -4, -1, 2, 5, 8, 11, 14, 17]

Expected	Your Output
[[-10, -7, 17], [-10, -4, 14], [-10, -1, 11], [-10, 2, 8], [-7, -4, 11], [-7, -1, 8], [-7, 2, 5], [-4, -1, 5]]	none

Case 5

nums: [-10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

Expected	Your Output
[[-10, 1, 9], [-10, 2, 8], [-10, 3, 7], [-10, 4, 6], [-9, 0, 9], [-9, 1, 8], [-9, 2, 7], [-9, 3, 6], [-9, 4, 5], [-8, -1, 9], [-8, 0, 8], [-8, 1, 7], [-8, 2, 6], [-8, 3, 5], [-7, -2, 9], [-7, -1, 8], [-7, 0, 7], [-7, 1, 6], [-7, 2, 5], [-7, 3, 4], [-6, -3, 9], [-6, -2, 8], [-6, -1, 7], [-6, 0, 6], [-6, 1, 5], [-6, 2, 4], [-5, -4, 9], [-5, -3, 8], [-5, -2, 7], [-5, -1, 6], [-5, 0, 5], [-5, 1, 4], [-5, 2, 3], [-4, -3, 7], [-4, -2, 6], [-4, -1, 5], [-4, 0, 4], [-4, 1, 3], [-3, -2, 5], [-3, -1, 4], [-3, 0, 3], [-3, 1, 2], [-2, -1, 3], [-2, 0, 2], [-1, 0, 1]]	none

Reference Solution

```typst
#let solution-ref(nums) = {
  let nums = nums.sorted()
  let n = nums.len()
  let ans = ()
  for i in range(n) {
    if i > 0 and nums.at(i) == nums.at(i - 1) {
      continue
    }
    let l = i + 1
    let r = n - 1
    while l < r {
      let sum = nums.at(i) + nums.at(l) + nums.at(r)
      if sum < 0 {
        l += 1
      } else if sum > 0 {
        r -= 1
      } else {
        ans.push((nums.at(i), nums.at(l), nums.at(r)))
        while l < r and nums.at(l) == nums.at(l + 1) {
          l += 1
        }
        while l < r and nums.at(r) == nums.at(r - 1) {
          r -= 1
        }
        l += 1
        r -= 1
      }
    }
  }
  ans
}
```

0016. 3Sum Closest

Given an integer array `nums` of length `n` and an integer `target`, find three integers in `nums` such that the sum is closest to `target`.

Return the sum of the three integers.

You may assume that each input would have exactly one solution.

Test Results

Case 1

nums: [-1, 2, 1, -4]
target: 1

Expected	Your Output
2	none

Case 2

nums: [0, 0, 0]
target: 1

Expected	Your Output
0	none

Case 3

nums: [0, 1, 1]
target: 2

Expected	Your Output
2	none

Case 4

nums: [-10, -7, -4, -1, 2, 5, 8, 11, 14, 17]
target: 20

Expected	Your Output
21	none

Case 5

nums: [-10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
target: 30

Expected	Your Output
24	none

Reference Solution

```typst
#let solution-ref(nums, target) = {
  let nums = nums.sorted()
  let n = nums.len()
  let ans = 2147483647
  for i in range(n) {
    if i > 0 and nums.at(i) == nums.at(i - 1) {
      continue
    }
    let l = i + 1
    let r = n - 1
    while l < r {
      let sum = nums.at(i) + nums.at(l) + nums.at(r)
      if sum < target {
        l += 1
      } else if sum > target {
        r -= 1
      } else {
        return target
      }
      if calc.abs(sum - target) < calc.abs(ans - target) {
        ans = sum
      }
    }
  }
  ans
}
```

0017. Letter Combinations of a Phone Number

Given a string containing digits from `2-9` inclusive, return all possible letter combinations that the number could represent. Return the answer in **any order**.

A mapping of digits to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.

Test Results

Case 1

digits: "23"

Expected	Your Output
["ad", "bd", "cd", "ae", "be", "ce", "af", "bf", "cf"]	none

Case 2

digits: ""

Expected	Your Output
[]	none

Case 3

digits: "2"

Expected	Your Output
["a", "b", "c"]	none

Reference Solution

```typst
#let solution-ref(digits) = {
  let digits = digits.clusters()
  let d = (
    "2": ("a", "b", "c"),
    "3": ("d", "e", "f"),
    "4": ("g", "h", "i"),
    "5": ("j", "k", "l"),
    "6": ("m", "n", "o"),
    "7": ("p", "q", "r", "s"),
    "8": ("t", "u", "v"),
    "9": ("w", "x", "y", "z"),
  )
  let ans = ("",)
  for c in digits {
    let nxt = ()
    for nc in d.at(c) {
      for s in ans {
        nxt.push(s + nc)
      }
    }
    ans = nxt
  }

  if ans == ("",) { () } else { ans }
}
```

0018. 4Sum

Given an array `nums` of `n` integers, return an array of all the unique quadruplets `[nums[a], nums[b], nums[c], nums[d]]` such that:

- `0 <= a, b, c, d < n`
- `a`, `b`, `c`, and `d` are **distinct**.
- `nums[a] + nums[b] + nums[c] + nums[d] == target`

You may return the answer in **any order**.

Test Results

Case 1

nums: [1, 0, -1, 0, -2, 2]
target: 0

Expected	Your Output
[[-2, -1, 1, 2], [-2, 0, 0, 2], [-1, 0, 0, 1]]	none

Case 2

nums: [2, 2, 2, 2]
target: 8

Expected	Your Output
[[2, 2, 2, 2]]	none

Case 3

nums: [-5, -4, -3, -2, -1, 0, 1, 2, 3, 4]
target: 3

Expected	Your Output
[[-5, 1, 3, 4], [-4, 0, 3, 4], [-4, 1, 2, 4], [-3, -1, 3, 4], [-3, 0, 2, 4], [-3, 1, 2, 3], [-2, -1, 2, 4], [-2, 0, 1, 4], [-2, 0, 2, 3], [-1, 0, 1, 3]]	none

Reference Solution

```typst
#let solution-ref(nums, target) = {
  let nums = nums.sorted()
  let n = nums.len()
```

```
    let ans = ()
    for i in range(n) {
      if i > 0 and nums.at(i) == nums.at(i - 1) {
        continue
      }
      for j in range(i + 1, n) {
        if j > i + 1 and nums.at(j) == nums.at(j - 1) {
          continue
        }
        let l = j + 1
        let r = n - 1
        while l < r {
          let sum = nums.at(i) + nums.at(j) + nums.at(l) + nums.at(r)
          if sum < target {
            l += 1
          } else if sum > target {
            r -= 1
          } else {
            ans.push((nums.at(i), nums.at(j), nums.at(l), nums.at(r)))
            while l < r and nums.at(l) == nums.at(l + 1) {
              l += 1
            }
            while l < r and nums.at(r) == nums.at(r - 1) {
              r -= 1
            }
            l += 1
            r -= 1
          }
        }
      }
    }
    ans
}
```

0019. Remove Nth Node From End of List

Given the head of a linked list, remove the nth node from the end of the list and return its head.

Test Results

Case 1

head: 1→2→3→4→5
n: 2

Expected	Your Output
1→2→3→5	none

Case 2

head: 1
n: 1

Expected	Your Output
∅	none

Case 3

head: 1→2
n: 1

Expected	Your Output
1	none

Reference Solution

```typst
#let solution-ref(head, n) = {
  let values = ()
  let node = head
  while node.next != none {
    values.push(node.val)
    node = node.next
  }

  let remove-index = values.len() - n
  let filtered = ()
```

```
  for i in range(values.len()) {
    if i != remove-index {
      filtered.push(values.at(i))
    }
  }
  linkedlist(filtered)
}
```

0020. Valid Parentheses

Given a string `s` containing just the characters `'('`, `')'`, `'{'`, `'}'`, `'['` and `']'`, determine if the input string is valid.

An input string is valid if:

1. Open brackets must be closed by the same type of brackets.
2. Open brackets must be closed in the correct order.
3. Every close bracket has a corresponding open bracket of the same type.

- `s` consists of parentheses only `'()[]{}'`.

Test Results

Case 1

s: "()"

Expected	Your Output
true	none

Case 2

s: "()[]{}"

Expected	Your Output
true	none

Case 3

s: "(]"

Expected	Your Output
false	none

Case 4

s: "([])"

Expected	Your Output
true	none

Case 5

s: "([)]"

Expected	Your Output
false	none

Reference Solution

```typst
#let solution-ref(s) = {
  // Performance Note: Use dict instead of array for stack
  //
  // Why dict? Array's slice(0, -1) is O(n) per pop → O(n²) total.
  // Dict + pointer avoids copies: insert O(1), decrement O(1).
  //
  // Stack pattern: dict + top pointer
  //   - push: stack.insert(str(top), val); top += 1
  //   - pop:  top -= 1
  //   - peek: stack.at(str(top - 1))
  //   - empty: top == 0
  let stack = (:)
  let top = 0
  let pairs = (
    ")": "(",
    "]": "[",
    "}": "{",
  )

  for char in s {
    if char in ("(", "[", "{") {
      stack.insert(str(top), char)
      top += 1
    } else if char in pairs {
      if top == 0 or stack.at(str(top - 1)) != pairs.at(char) {
        return false
      }
      top -= 1
    }
  }

  top == 0
}
```

0021. N-Queens

The **n-queens** puzzle is the problem of placing n queens on an $n \times n$ chessboard such that no two queens attack each other.

Given an integer n, return *all distinct solutions* to the **n-queens puzzle**. You may return the answer in **any order**.

Each solution contains a distinct board configuration of the n-queens' placement, where `'Q'` and `'.'` both indicate a queen and an empty space, respectively.

Test Results

Case 1

n: 1

Expected	Your Output
Q	none

Case 2

n: 2

Expected	Your Output
[]	none

Case 3

n: 4

Expected	Your Output
. Q . . . . . Q Q . . . . . Q . . . Q . Q . . . . . . Q . Q . .	none

Reference Solution

```typst
#let solution-ref(n) = {
  if n == 0 {
    return ((),)
  }

  let build-board(positions) = {
    range(n).map(r => range(n)
      .map(c => if c == positions.at(r) { "Q" } else { "." })
      .join())
  }

  let is-safe(positions, row, col) = {
    for r in range(row) {
      let c = positions.at(r)
      if c == col or calc.abs(r - row) == calc.abs(c - col) {
        return false
      }
    }
    true
  }

  let solve(positions, row) = {
    if row == n {
      return (build-board(positions),)
    }

    let ret = ()
    for col in range(n) {
      if not is-safe(positions, row, col) {
        continue
      }
      let new-pos = positions.map(x => x)
      new-pos.push(col)
      for board in solve(new-pos, row + 1) {
        ret.push(board)
      }
    }
    ret
  }

  solve((), 0)
}
```