1. Remove Element

Python
```
def remove_element(nums, val):
  """
  In-place removal of elements with value `val` from array `nums`.

  Args:
      nums: An array of integers.
      val: The value to remove.

  Returns:
      The new length of the array after removing elements.
  """
  i = 0
  for num in nums:
    if num != val:
      nums[i] = num
      i += 1
  return i
```

2. Valid Sudoku

Python
```
def is_valid_sudoku(board):
  """
  Checks if a Sudoku board is valid.

  Args:
      board: A list of lists representing the Sudoku board.

  Returns:
      True if the board is valid, False otherwise.
  """
  rows = cols = boxes = [[False] * 9 for _ in range(9)]
  for row in range(9):
    for col in range(9):
      if board[row][col] != ".":
        box_index = (row // 3 ) * 3 + col // 3
        if rows[row][ord(board[row][col]) - 49] or
cols[col][ord(board[row][col]) - 49] or
boxes[box_index][ord(board[row][col]) - 49]:
          return False
        rows[row][ord(board[row][col]) - 49] =
cols[col][ord(board[row][col]) - 49] =
boxes[box_index][ord(board[row][col]) - 49] = True
  return True
```

3. Count and Say

Python
```
def count_and_say(n):
  """
  Generates the nth term of the count-and-say sequence.

  Args:
      n: The term number.
```

```
  Returns:
      The nth term of the count-and-say sequence.
  """
  if n == 1:
    return "1"
  prev = count_and_say(n - 1)
  result = ""
  count = 1
  for i in range(1, len(prev)):
    if prev[i] == prev[i-1]:
      count += 1
    else:
      result += str(count) + prev[i-1]
      count = 1
  result += str(count) + prev[-1]
  return result
```

40. Combination Sum II

Python

```
def combination_sum2(candidates, target):
  """
  Finds all unique combinations of candidates that sum to target (no
duplicates allowed).

  Args:
      candidates: A list of integers (candidates).
      target: The target sum.

  Returns:
      A list of lists containing unique combinations that sum to target.
  """
  results = []
  def backtrack(curr, start):
    if sum(curr) == target:
      results.append(list(curr))
      return
    if sum(curr) > target or start >= len(candidates):
      return
    for i in range(start, len(candidates)):
      curr.append(candidates[i])
      backtrack(curr, i + 1)  # Don't pick the same element again
      curr.pop()
  backtrack([], 0)
  return results
```

53. Maximum Subarray

Python

```
def max_subarray(nums):
  """
  Finds the subarray with the maximum sum in an integer array.

  Args:
      nums: An array of integers.

  Returns:
```

```
    The maximum sum of a subarray.
  """
  current_sum = max_sum = nums[0]
  for num in nums[1:]:
    current_sum = max(num, current_sum + num)
    max_sum = max(max_sum, current_sum)
  return max_sum
```

Length of Last Word

Python

```
def length_of_last_word(s):
  """
  Finds the length of the last word in a string.

  Args:
      s: A string containing words and spaces.

  Returns:
      The length of the last word.
  """
  words = s.strip().split()
  return len(words[-
```

37. Sudoku Solver

Python

```
def solve_sudoku(board):
  """
  Solves a Sudoku puzzle using backtracking.

  Args:
      board: A list of lists representing the Sudoku board.

  Returns:
      True if a solution is found, False otherwise.
  """
  def backtrack(row, col):
    if col == 9:
      return solve_sudoku(board, row + 1, 0)  # Move to next row
    if board[row][col] != ".":
      return backtrack(row, col + 1)

    for num in range(1, 10):
      if is_valid(board, row, col, num):
        board[row][col] = str(num)
        if backtrack(row, col + 1):
          return True
        board[row][col] = "."

    return False  # Backtrack if no valid number found

  def is_valid(board, row, col, num):
    for i in range(9):
      if board[row][i] == str(num) or board[i][col] == str(num):
        return False
    start_row = row // 3 * 3
    start_col = col // 3 * 3
```

```
    for i in range(start_row, start_row + 3):
      for j in range(start_col, start_col + 3):
        if board[i][j] == str(num):
          return False
    return True

  if not solve_sudoku(board, 0, 0):
    return False
  return True
```

39. Combination Sum

Python

```
def combination_sum(candidates, target):
  """
  Finds all unique combinations of candidates that sum to target
(duplicates allowed).

  Args:
      candidates: A list of integers (candidates).
      target: The target sum.

  Returns:
      A list of lists containing unique combinations that sum to target.
  """
  results = []
  def backtrack(curr, start):
    if sum(curr) == target:
      results.append(list(curr))
      return
    if sum(curr) > target:
      return
    for i in range(start, len(candidates)):
      curr.append(candidates[i])
      backtrack(curr, start)  # Can pick the same element again
      curr.pop()
  backtrack([], 0)
  return results
```

46. Permutations

Python

```
def permute(nums):
  """
  Finds all permutations of a list of numbers.

  Args:
      nums: A list of integers.

  Returns:
      A list of lists containing all permutations.
  """
  results = []
  def backtrack(curr):
    if len(curr) == len(nums):
      results.append(list(curr))
      return
    for num in nums:
      if num not in curr:
```

```
        curr.append(num)
        backtrack(curr)
        curr.pop()
  backtrack([])
  return results
```

67. Add Binary

Python

```
def add_binary(a, b):
  """
  Adds two binary strings and returns the sum in binary.

  Args:
      a: A binary string.
      b: A binary string.

  Returns:
      The sum of the two binary strings in binary.
  """
  carry = 0
  result = ""
  a = a[::-1]
  b = b[::-1]
  i = 0
  while i < len(a) or i < len(b) or carry:
    val1 = int(a[i]) if i < len(a) else 0
    val2 = int(b[i]) if i < len(b) else 0
    sum = val1 + val2 + carry
    carry = sum // 2
    result += str(sum % 2)
    i += 1
  return result[::-1]
```