


33. Find All Permutations of a String

```
from itertools import permutations

def find_permutations(s):
    return [''.join(p) for p in permutations(s)]

# Example usage
print(find_permutations("abc"))
```

 ['abc', 'acb', 'bac', 'bca', 'cab', 'cba']

34. N-th Fibonacci Number (Dynamic Programming)

```
def fibonacci(n):
    if n <= 1:
        return n
    fib = [0, 1]
    for i in range(2, n + 1):
        fib.append(fib[i - 1] + fib[i - 2])
    return fib[n]


# Example usage
print(fibonacci(10))
```

35. Find Duplicates in a List

```
from collections import Counter

def find_duplicates(lst):
    count = Counter(lst)
    return [item for item, freq in count.items() if freq > 1]

# Example usage
print(find_duplicates([1, 2, 3, 2, 4, 5, 6, 6]))
```

 [2, 6]

36. Longest Increasing Subsequence (LIS)

```
def length_of_lis(nums):
    if not nums:
        return 0
    dp = [1] * len(nums)
    for i in range(len(nums)):
```

```

        for j in range(i):
            if nums[i] > nums[j]:
                dp[i] = max(dp[i], dp[j] + 1)
        return max(dp)

# Example usage
print(length_of_lis([10, 9, 2, 5, 3, 7, 101, 18]))

```

↩ 4

37. Find K Largest Elements

```

import heapq

def find_k_largest(lst, k):
    return heapq.nlargest(k, lst)

# Example usage
print(find_k_largest([3, 1, 5, 12, 2, 11], 3))

```

↩ [12, 11, 5]

38. Rotate Matrix

```

def rotate_matrix(matrix):
    return [list(row) for row in zip(*matrix[::-1])]

# Example usage
matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]
rotated = rotate_matrix(matrix)
for row in rotated:
    print(row)

```

↩ [7, 4, 1]
[8, 5, 2]
[9, 6, 3]

39. Sudoku Validator

```

def is_valid_sudoku(board):
    def is_valid_block(block):
        block = [num for num in block if num != "."]
        return len(block) == len(set(block))

```

```

    for row in board:
        if not is_valid_block(row):
            return False

    for col in zip(*board):
        if not is_valid_block(col):
            return False

    for i in range(0, 9, 3):
        for j in range(0, 9, 3):
            block = [board[x][y] for x in range(i, i + 3) for y in range(j, j + 3)]
            if not is_valid_block(block):
                return False

    return True

# Example usage
sudoku_board = [
    ["5", "3", ".", ".", "7", ".", ".", ".", "."],
    ["6", ".", ".", "1", "9", "5", ".", ".", "."],
    [".", "9", "8", ".", ".", ".", ".", "6", "."],
    ["8", ".", ".", ".", "6", ".", ".", ".", "3"],
    ["4", ".", ".", "8", ".", "3", ".", ".", "1"],
    ["7", ".", ".", ".", "2", ".", ".", ".", "6"],
    [".", "6", ".", ".", ".", ".", "2", "8", "."],
    [".", ".", ".", "4", "1", "9", ".", ".", "5"],
    [".", ".", ".", ".", "8", ".", ".", "7", "9"]
]
print(is_valid_sudoku(sudoku_board))

```

⇒ True

40. Virtual Stock Market Simulator

```

import random

class StockMarketSimulator:
    def __init__(self, stocks):
        self.stocks = {stock: 100 for stock in stocks} # Initial price

    def update_prices(self):
        for stock in self.stocks:
            change = random.uniform(-5, 5) # Random price fluctuation
            self.stocks[stock] = max(1, self.stocks[stock] + change)

    def display_prices(self):
        for stock, price in self.stocks.items():
            print(f"{stock}: ${price:.2f}")

# Example usage
simulator = StockMarketSimulator(["AAPL", "GOOGL", "TSLA"])
simulator.update_prices()
simulator.display_prices()

```



AAPL: \$97.21
GOOGL: \$95.77
TSLA: \$100.13