**Day 1: Arrays - Basics and Prefix/Suffix**

**🧮 Two Sum**

**Problem Statement:**  
Given a list of item prices, find indices of two items whose combined price equals a customer's gift card value.

**Sample Input:**  
nums = [2, 7, 11, 15], target = 9

**Expected Output:**  
[0, 1]  
Explanation: The items at index 0 and 1 add up to 9 (2 + 7 = 9).

**📈 Maximum Subarray (Kadane’s Algorithm)**

**Problem Statement:**  
Find the contiguous period of maximum net income from daily financial logs.

**Sample Input:**  
nums = [-2, 1, -3, 4, -1, 2, 1, -5, 4]

**Expected Output:**  
6  
Explanation: Subarray [4, -1, 2, 1] yields the highest sum.

**🚚 Move Zeroes**

**Problem Statement:**  
Rearrange a delivery queue where zeros indicate unavailable trucks; push them to the end without disrupting the rest.

**Sample Input:**  
nums = [0, 1, 0, 3, 12]

**Expected Output:**  
[1, 3, 12, 0, 0]

**💹 Best Time to Buy and Sell Stock**

**Problem Statement:**  
Identify the best day to buy and the best day to sell for highest profit from historical stock prices.

**Sample Input:**  
prices = [7, 1, 5, 3, 6, 4]

**Expected Output:**  
5  
Explanation: Buy at 1 and sell at 6.

**⚖️ Find Pivot Index**

**Problem Statement:**  
Determine the index at which the workload to its left equals the workload to its right.

**Sample Input:**  
nums = [1, 7, 3, 6, 5, 6]

**Expected Output:**  
3  
Explanation: 1 + 7 + 3 = 11, 5 + 6 = 11.

**🧴 Product of Array Except Self**

**Problem Statement:**  
For each brand, calculate its relative contribution by excluding itself from the total.

**Sample Input:**  
nums = [1, 2, 3, 4]

**Expected Output:**  
[24, 12, 8, 6]

**🔁 Merge Intervals**

**Problem Statement:**  
Combine overlapping appointment slots to simplify scheduling.

**Sample Input:**  
intervals = [[1, 3], [2, 6], [8, 10], [15, 18]]

**Expected Output:**  
[[1, 6], [8, 10], [15, 18]]

**📥 Insert Interval**

**Problem Statement:**  
Insert a new booking and merge overlapping intervals in an existing schedule.

**Sample Input:**  
intervals = [[1, 3], [6, 9]], new\_interval = [2, 5]

**Expected Output:**  
[[1, 5], [6, 9]]

**🌀 Spiral Matrix**

**Problem Statement:**  
Retrieve values from a 2D grid in spiral order, like reading a blueprint from the outermost layer inward.

**Sample Input:**  
matrix = [[1,2,3],[4,5,6],[7,8,9]]

**Expected Output:**  
[1, 2, 3, 6, 9, 8, 7, 4, 5]

**⛔ Set Matrix Zeroes**

**Problem Statement:**  
If any data point in a table is corrupted (zero), reset its row and column entirely.

**Sample Input:**  
matrix = [[1,1,1],[1,0,1],[1,1,1]]

**Expected Output:**  
[[1,0,1],[0,0,0],[1,0,1]]

**🔄 Rotate Image**

**Problem Statement:**  
Design a tool that rotates an image represented as a 2D matrix by 90° clockwise — useful for layout and orientation tasks.

**Sample Input:**  
matrix = [[1,2,3], [4,5,6], [7,8,9]]

**Expected Output:**  
[[7,4,1], [8,5,2], [9,6,3]]  
Explanation: The matrix is rotated layer by layer.

**🎯 Subarray Sum Equals K**

**Problem Statement:**  
In transaction logs, count how many periods (contiguous subarrays) have total spending equal to a budget target.

**Sample Input:**  
nums = [1, 1, 1], k = 2

**Expected Output:**  
2  
Explanation: Two subarrays [1,1] at positions [0,1] and [1,2] meet the condition.

**📆 Longest Consecutive Sequence**

**Problem Statement:**  
Track the longest streak of daily activity, such as attendance, step logging, or app usage.

**Sample Input:**  
nums = [100, 4, 200, 1, 3, 2]

**Expected Output:**  
4  
Explanation: Longest sequence is [1,2,3,4].

**🌬️ Sliding Window Maximum**

**Problem Statement:**  
Monitor and report the highest reading (e.g., temperature, bandwidth, CPU usage) in each fixed-duration window.

**Sample Input:**  
nums = [1,3,-1,-3,5,3,6,7], k = 3

**Expected Output:**  
[3,3,5,5,6,7]  
Explanation: Each 3-element window’s maximum is extracted.

**🎨 Sort Colors (Dutch National Flag)**

**Problem Statement:**  
Sort an inventory with three status flags (e.g. urgent=2, normal=1, low=0) to streamline priority processing.

**Sample Input:**  
nums = [2, 0, 2, 1, 1, 0]

**Expected Output:**  
[0, 0, 1, 1, 2, 2]

**🔢 Count Inversions**

**Problem Statement:**  
Analyze disruptions in queue or priority order — count how many pairwise swaps are needed to correct it.

**Sample Input:**  
nums = [2, 3, 8, 6, 1]

**Expected Output:**  
5  
Explanation: Inversions include (2,1), (3,1), (8,6), (8,1), (6,1).

**🚀 Next Permutation**

**Problem Statement:**  
Build system-generated version numbers or ID sequences that move to the next lexicographical order.

**Sample Input:**  
nums = [1,2,3]

**Expected Output:**  
[1,3,2]  
Explanation: [1,3,2] is the next closest permutation after [1,2,3].

**🎲 3Sum**

**Problem Statement:**  
Check if any three contributors can offset a financial loss or debt exactly to zero.

**Sample Input:**  
nums = [-1, 0, 1, 2, -1, -4]

**Expected Output:**  
[[-1, -1, 2], [-1, 0, 1]]  
Explanation: These triplets sum to zero.

**🎯 4Sum**

**Problem Statement:**  
In budget planning, find combinations of four departmental spends that match the overall cap.

**Sample Input:**  
nums = [1, 0, -1, 0, -2, 2], target = 0

**Expected Output:**  
[[-2,-1,1,2], [-2,0,0,2], [-1,0,0,1]]

**🗳️ Majority Element**

**Problem Statement:**  
Identify the most dominant choice in polling data — the one selected by more than half of the voters.

**Sample Input:**  
nums = [3, 2, 3]

**Expected Output:**  
3  
Explanation: The number 3 appears more than ⌊n/2⌋ times.