**Day 4: Two Sum & Variants**

**🧮 1. Two Sum**

**Problem Statement:**  
In an expense tracking app, identify two transactions whose total matches a suspicious flagged amount.  
**Sample Input:**  
nums = [2, 7, 11, 15], target = 9  
**Expected Output:**  
[0, 1]

**🔺 2. Three Sum**

**Problem Statement:**  
Find combinations of three financial offsets in a ledger that neutralize each other (sum to zero).  
**Sample Input:**  
nums = [-1, 0, 1, 2, -1, -4]  
**Expected Output:**  
[[-1, -1, 2], [-1, 0, 1]]

**🎯 3. Four Sum**

**Problem Statement:**  
Identify four expenses that together match a reimbursement 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]]

**🔍 4. Two Sum II – Sorted Array**

**Problem Statement:**  
Efficiently search a sorted invoice list for two amounts that match a budget limit.  
**Sample Input:**  
numbers = [2, 7, 11, 15], target = 9  
**Expected Output:**  
[1, 2] (1-indexed)

**🌳 5. Two Sum in BST**

**Problem Statement:**  
Given a binary search tree of account balances, detect if any two balances sum to a flagged threshold.  
**Sample Input:**  
BST = [5, 3, 6, 2, 4, null, 7], k = 9  
**Expected Output:**  
True

**🔗 6. Count Pairs With Given Sum**

**Problem Statement:**  
Count how many unique pairs of donations together match a grant target.  
**Sample Input:**  
arr = [1, 5, 7, -1, 5], target = 6  
**Expected Output:**  
3

**🧾 7. Longest Subarray With Zero Sum**

**Problem Statement:**  
In budget analysis, find the longest period where inflows and outflows neutralize each other.  
**Sample Input:**  
nums = [15, -2, 2, -8, 1, 7, 10, 23]  
**Expected Output:**  
5

**📊 8. Subarray Sum Equals K**

**Problem Statement:**  
Count the number of spending streaks that perfectly hit a weekly budget goal.  
**Sample Input:**  
nums = [1, 1, 1], k = 2  
**Expected Output:**  
2

**🧮 9. K-Diff Pairs in Array**

**Problem Statement:**  
Track how many expense pairs differ by exactly k — helpful in variance checks.  
**Sample Input:**  
nums = [3, 1, 4, 1, 5], k = 2  
**Expected Output:**  
2

**🔍 10. Find All Duplicates**

**Problem Statement:**  
Detect duplicate entries in a data stream like transaction IDs or user logins.  
**Sample Input:**  
nums = [4, 3, 2, 7, 8, 2, 3, 1]  
**Expected Output:**  
[2, 3]

**🔁 11. Continuous Subarray Sum**

**Problem Statement:**  
Detect if a continuous sequence of transactions sums to a multiple of k — often used in fraud detection or batch settlement tracking.  
**Sample Input:**  
nums = [23, 2, 4, 6, 7], k = 6  
**Expected Output:**  
True

**🧬 12. Longest Consecutive Sequence**

**Problem Statement:**  
Identify the longest series of consecutive days with logged activities — vital for loyalty programs or habit tracking.  
**Sample Input:**  
nums = [100, 4, 200, 1, 3, 2]  
**Expected Output:**  
4  
*Explanation: The streak is [1, 2, 3, 4].*

**🎵 13. Pairs of Songs With Durations Divisible by 60**

**Problem Statement:**  
In a music streaming app, count how many song pairings result in a playlist duration perfectly divisible by 60 seconds.  
**Sample Input:**  
time = [30, 20, 150, 100, 40]  
**Expected Output:**  
3

**💡 14. Sum of Two Integers (Bit Manipulation)**

**Problem Statement:**  
Emulate integer addition at the lowest computational level — common in system-level coding and embedded systems.  
**Sample Input:**  
a = 1, b = 2  
**Expected Output:**  
3

**⚖️ 15. Equal Zero One Subarray**

**Problem Statement:**  
Evaluate binary feedback (0s and 1s) to find the longest balanced segment — useful in surveys or signal analysis.  
**Sample Input:**  
nums = [0, 1, 0, 0, 1, 1, 0]  
**Expected Output:**  
6

**🎯 16. Pair With Given Difference**

**Problem Statement:**  
Check if two product prices differ exactly by a fixed value — applicable in price matching algorithms.  
**Sample Input:**  
arr = [1, 8, 30, 40, 100], diff = 60  
**Expected Output:**  
True

**🧩 17. Count Quadruplets That Sum to Target**

**Problem Statement:**  
In a combinatorial database, count all unique groups of four items whose values sum up to a given target — helpful in pricing bundles or set matching.  
**Sample Input:**  
nums = [1, 2, 3, 6], target = 10  
**Expected Output:**  
1

**✖️ 18. XOR Pairs**

**Problem Statement:**  
In cybersecurity, count how many pairings of data produce a specific XOR hash — used in encryption and checksum validation.  
**Sample Input:**  
arr = [1, 2, 3, 4], target = 6  
**Expected Output:**  
1

**🧮 19. Min Operations to Make Array Sum Zero**

**Problem Statement:**  
Calculate minimum adjustment operations to neutralize total spending — used in audit cleanup or balancing algorithms.  
**Sample Input:**  
nums = [1, 1, 2, 2, 3, 3]  
**Expected Output:**  
12

**🔐 20. Max Number of K-Sum Pairs**

**Problem Statement:**  
In a gifting app, match users so that their contributions total to k — maximizing the number of such unique pairings.  
**Sample Input:**  
nums = [1, 2, 3, 4], k = 5  
**Expected Output:**  
2  
*Explanation: Pairs are (1,4) and (2,3).*