**🚀 Real-World Practice Problems for Junior Developer Interviews**

**🟢 Easy (warm-up / junior screening level)**

These test fundamentals: arrays, objects, loops, conditionals.

1. **Running Balance** (Banking)
   * Input: array of {type: "deposit"/"withdrawal", amount}
   * Output: array of running balances after each transaction.
   * Skills: conditionals, accumulation.
2. **Most Recent Order per Customer**
   * Input: orders = [{customerId, date, total}]
   * Output: latest order for each customer.
   * Skills: grouping, comparing dates.
   * Hint: Map or object with customerId as key.
3. **Word Frequency Counter** (like logs, tweets, chat messages)
   * Input: string "hello world hello"
   * Output: {hello: 2, world: 1}
   * Skills: reduce / object counting.
4. **Unique IDs**
   * Input: [1, 2, 2, 3, 4, 4, 5]
   * Output: [1, 2, 3, 4, 5]
   * Skills: Set / filtering duplicates.
5. **Parking Lot Tracker (your past one)**
   * Extend: support multiple floors with different spot counts.
   * Add feature: search car by license.
   * Level-up: add maxHours for cars and charge fee.

**🟡 Medium (typical interview / real-world logic)**

These mimic actual business problems. You’ll see this type a lot.

1. **Inventory Manager**
   * Input: transactions = [{item: "apple", qty: +10}, {item: "apple", qty: -3}]
   * Output: {apple: 7}
   * Skills: reduce, handling negative stock, objects.
2. **Task List with Status**
   * Build a small class TaskManager.
     + addTask(name)
     + completeTask(id)
     + listTasks(status) → "all", "completed", "pending"
   * Skills: OOP, arrays, filtering.
3. **Merge Two Sorted Arrays**
   * Input: [1,3,5] and [2,4,6]
   * Output: [1,2,3,4,5,6]
   * Skills: two-pointer technique, efficiency.
4. **Pagination**
   * Input: 100 items, page size = 10, page = 3
   * Output: items 21–30
   * Skills: slicing, indices, reusability.
5. **Simple Search by Keyword**

* Input: tasks = ["buy milk", "call mom", "fix bug"]
* Query = "call" → ["call mom"]
* Skills: string includes, filtering.

**🔴 Hard (system thinking / scaling / “how would you design”)**

These come up in **final rounds** to test problem-solving and thinking out loud.

1. **Event Booking System**

* Users can book seats (A1, A2, A3...).
* Prevent double-booking the same seat.
* Cancel booking.
* Skills: sets/maps, consistency, edge cases.

1. **Chat Room Simulator**

* Users join/leave.
* Keep track of active users.
* Broadcast message = show [user]: message to all active.
* Skills: OOP, arrays/maps.

1. **URL Shortener (like bit.ly)**

* Input: long URL → short random code (6 chars).
* Store mapping {code → url}.
* Retrieve long URL when code is given.
* Skills: maps, random generation.

1. **Rate Limiter** (real backend question)

* Only allow max 5 requests per user per minute.
* Block if exceeded.
* Skills: timestamps, arrays, clean old entries.

1. **File Upload Tracker**

* Given chunks:
  + {id: 1, chunk: 1}, {id: 1, chunk: 2} …
* Reconstruct full file per id.
* Skills: grouping, completeness check.

**📝 Practice Interview Questions (Ask yourself out loud)**

For every problem, try to answer like in a real interview:

1. “How will you store the data?” → arrays, objects, maps.
2. “What happens if input is invalid?” → guard clauses.
3. “How does this scale to 10k?” → efficient loops, maps instead of nested loops.
4. “How would you persist data?” → right now in memory, later DB.
5. “How would you test this?” → console logs, unit tests with sample inputs.
6. “Tradeoffs?” → readability vs. performance.
7. “Can you extend it?” → add new feature easily?

🔥 Here’s what I recommend:

* Pick **1 problem per day**.
* Solve it once **cleanly**, then try again the next day **without looking at your notes** (simulate pressure).
* After coding, answer 2–3 of the **interview-style questions** out loud.

If you grind these, in your next interview you’ll look *way sharper*.