Java Collections Problems

🔷 1. List (ArrayList, LinkedList, Vector)

❓ Problem	🔍 Concept
🔷 Remove duplicates from a list while preserving order	Use `LinkedHashSet` or check with `contains()`
🔷 Find the second largest element in a list	Sort or traverse twice
🔷 Reverse a list without using Collections.reverse()	Loop and swap
🔷 Merge two sorted lists	Like merge step in merge sort
🔷 Detect a cycle in a linked list (use LinkedList logic)	Use Floyd's Cycle Algorithm
🔷 Convert array to list and list to array	`Arrays.asList()` and `list.toArray()`
🔷 Rotate a list to the right by k steps	Modulo + slicing logic

🔷 2. Set (HashSet, LinkedHashSet, TreeSet)

❓ Problem	🔍 Concept
🔷 Find the union and intersection of two sets	`addAll()` and `retainAll()`
🔷 Find all distinct characters in a string	Store in `Set<Character>`
🔷 Find duplicate elements in an array	Check `set.contains()` before adding
🔷 Remove duplicates from a list	Convert to `Set` and back to `List`
🔷 Check if two strings are anagrams	Use `Set` and sorting or frequency maps
🔷 Count number of distinct words in a paragraph	Use `Set<String>` with `split()`
🔷 Sort a set of elements	Use `TreeSet`

🔷 3. Queue (LinkedList, PriorityQueue)

❓ Problem	🔍 Concept
🔷 Implement a queue using two stacks	Stack logic, enqueue/dequeue
🔷 Print binary numbers from 1 to N	Use Queue to simulate number building
🔷 Implement a circular queue	Array + modulo logic
🔷 Find first non-repeating character in a stream	Use Queue + frequency map
🔷 Merge K sorted arrays	Use `PriorityQueue` (min-heap)
🔷 Design a task scheduler with priorities	Use `PriorityQueue` with custom comparator
🔷 Process jobs in order with delay	Use `PriorityQueue` to simulate delay queue

🔷 4. Deque (ArrayDeque, LinkedList)

❓ Problem	🔍 Concept
🔷 Check if a string is a palindrome	Compare front and rear
🔷 Implement a sliding window max (fixed size k)	Deque stores indices
🔷 Design a stack with `getMin()` in O(1)	Use 2 stacks or a custom class
🔷 Implement browser forward/backward history	Use Deque to simulate stack in both directions
🔷 Reverse first K elements of a queue	Use Deque to rotate
🔷 Design LRU Cache	Deque + HashMap or `LinkedHashMap`

🔷 5. Map (HashMap, TreeMap, LinkedHashMap)

❓ Problem	🔍 Concept
🔷 Count frequency of each word in a string	Use `Map<String, Integer>`
🔷 Find the first non-repeating character	Map for freq + order
🔷 Group anagrams together	Use sorted word as key
🔷 Find top K frequent elements	Map for count + `PriorityQueue`
🔷 Sort a map by values	Convert to list + comparator

🔷 Two-sum problem	Store complement in Map
🔷 Find common elements with same frequency in two arrays	Map for freq in both, compare
🔷 LRU Cache implementation	Use `LinkedHashMap` with accessOrder = true
🔷 Design a phonebook	Use `Map<String, List<String>>`
🔷 Count character frequency and print sorted	TreeMap or sort entries by value

✅ Bonus: Mixed / Real-World Problems

❓ Problem	🔍 Use
🔷 Build a leaderboard with live rank updates	`TreeMap` or custom `PriorityQueue`
🔷 Word auto-suggestion system	`Map<Character, TrieNode>`
🔷 Implement undo-redo	Two stacks or two deques
🔷 Parenthesis validation	Use `Stack`, implemented via `Deque`
🔷 Stock span problem	Stack or Deque
🔷 Subarray sum equals K	Use `Map` to store cumulative sum counts