**🔤 Basic Level (Key-Value Pairs, Insertion, Access)**

1. **Frequency Counter**  
   Write a program to count the frequency of each character in a given string using a map.  
   *Input:* "hello" → *Output:* {h:1, e:1, l:2, o:1}
2. **Word Count in a Sentence**  
   Given a sentence, count how many times each word appears.  
   *Input:* "this is a test this is fun" → *Output:* {this:2, is:2, a:1, test:1, fun:1}
3. **Check for Duplicates**  
   Write a function that checks if a list of numbers contains any duplicates using a map.  
   *Input:* [1, 2, 3, 4, 1] → *Output:* True

**🔁 Intermediate Level (Looping, Conditional Logic)**

1. **Find the First Non-Repeating Character**  
   Given a string, return the first character that doesn't repeat.  
   *Input:* "swiss" → *Output:* 'w'
2. **Group Anagrams**  
   Given a list of words, group all the anagrams together using a map.  
   *Input:* ["eat", "tea", "tan", "ate", "nat", "bat"]  
   *Output:* [["eat", "tea", "ate"], ["tan", "nat"], ["bat"]]
3. **Top K Frequent Elements**  
   Find the top k most frequent numbers in an array.  
   *Input:* [1, 1, 1, 2, 2, 3], k = 2 → *Output:* [1, 2]

**🔍 Advanced Level (Nested Maps, Combinations)**

1. **Longest Substring Without Repeating Characters**  
   Write a function that returns the length of the longest substring without repeating characters using a sliding window and a map.  
   *Input:* "abcabcbb" → *Output:* 3 (substring = "abc")
2. **Isomorphic Strings**  
   Given two strings, check if they are isomorphic (characters map consistently).  
   *Input:* "egg", "add" → *Output:* True  
   *Input:* "foo", "bar" → *Output:* False
3. **Subarray Sum Equals K**  
   Given an array of integers and a value k, find the total number of subarrays whose sum equals to k using prefix sums and maps.  
   *Input:* [1, 1, 1], k = 2 → *Output:* 2
4. **LRU Cache Simulation**  
   Implement an LRU (Least Recently Used) cache using a map and doubly linked list. Simulate insertion, retrieval, and deletion.