**Java Interview Coding Questions**

**1. Array and String Manipulation:**

* **Reverse a String**: Write a program to reverse a string without using the built-in reverse method.
* **Find Duplicate Elements in an Array**: Write a program to find duplicate elements in a given array.
* **Find the Largest and Second Largest Element in an Array**: Implement an efficient solution to find the two largest numbers in an array.
* **Rotate Array**: Rotate an array by k positions to the right or left.
* **Check if Two Strings are Anagrams**: Write a function to check if two strings are anagrams.
* **Find Missing Number in an Array**: Given an array containing n-1 integers between 1 to n, find the missing number.
* **Longest Substring Without Repeating Characters**: Find the length of the longest substring without repeating characters.

**2. Linked Lists:**

* **Reverse a Linked List**: Write a function to reverse a singly linked list.
* **Detect a Cycle in a Linked List**: Use Floyd’s Cycle Detection Algorithm to check if a linked list contains a cycle.
* **Find the Middle Element of a Linked List**: Write a function to find the middle of a linked list.
* **Merge Two Sorted Linked Lists**: Write a program to merge two sorted linked lists into a single sorted list.
* **Remove Nth Node from End of List**: Given a linked list, remove the N-th node from the end of the list and return the head.

**3. Recursion and Backtracking:**

* **Find Factorial Using Recursion**: Implement a function to find the factorial of a number using recursion.
* **Generate All Permutations of a String**: Write a function to generate all permutations of a given string.
* **N-Queens Problem**: Solve the N-Queens problem using backtracking.
* **Find All Subsets of a Set**: Write a function that returns all possible subsets of a set.

**4. Sorting and Searching:**

* **Binary Search**: Implement a binary search algorithm.
* **Merge Sort**: Write a program to perform merge sort on an array.
* **Quick Sort**: Implement quick sort on an array.
* **Find Kth Largest Element in an Array**: Given an array, find the k-th largest element using a sorting algorithm or heap.

**5. Dynamic Programming:**

* **Fibonacci Sequence**: Write a program to compute the nth Fibonacci number using dynamic programming.
* **Longest Common Subsequence**: Given two sequences, find the length of the longest subsequence present in both.
* **Coin Change Problem**: Given a set of coin denominations, write a program to find the minimum number of coins needed to make a given amount.
* **0/1 Knapsack Problem**: Solve the 0/1 knapsack problem using dynamic programming.

**6. Tree and Binary Tree Problems:**

* **Binary Tree Inorder, Preorder, and Postorder Traversal**: Write functions for inorder, preorder, and postorder traversal of a binary tree.
* **Check if a Binary Tree is Balanced**: Write a program to check if a binary tree is balanced.
* **Find the Lowest Common Ancestor in a Binary Tree**: Write a function to find the lowest common ancestor of two nodes in a binary tree.
* **Level Order Traversal of a Binary Tree**: Write a function to print the level order traversal of a binary tree.
* **Convert a Binary Search Tree to a Doubly Linked List**: Write a function to convert a binary search tree into a doubly linked list.

**7. Graphs:**

* **Depth First Search (DFS)**: Write a program to perform a DFS traversal of a graph.
* **Breadth First Search (BFS)**: Write a program to perform a BFS traversal of a graph.
* **Detect a Cycle in a Graph**: Implement a function to detect if a cycle exists in a directed or undirected graph.
* **Find Shortest Path in an Unweighted Graph**: Use BFS to find the shortest path between two nodes in an unweighted graph.

**8. Mathematical Problems:**

* **Check if a Number is Prime**: Write a program to check whether a number is prime or not.
* **Find GCD of Two Numbers**: Write a program to find the greatest common divisor (GCD) of two numbers using Euclid's algorithm.
* **Palindrome Number**: Write a program to check if a number is a palindrome without converting it to a string.
* **FizzBuzz Problem**: Print numbers from 1 to 100. For multiples of 3, print "Fizz"; for multiples of 5, print "Buzz"; and for multiples of both, print "FizzBuzz".

**9. Design Patterns and Best Practices:**

* **Singleton Design Pattern**: Implement the Singleton pattern in Java.
* **Producer-Consumer Problem**: Write a Java program to implement the producer-consumer problem using wait() and notify().
* **Thread-safe Singleton**: Implement a thread-safe version of the Singleton pattern.

**10. Miscellaneous Problem-Solving:**

* **LRU Cache**: Implement an LRU (Least Recently Used) Cache using a data structure like a HashMap and LinkedList.
* **Median of Two Sorted Arrays**: Write a function to find the median of two sorted arrays of different lengths.
* **Valid Parentheses**: Given a string containing just the characters (, ), {, }, [, and ], determine if the input string is valid.
* **Count Inversions in an Array**: Write a program to count the number of inversions required to sort an array.