1. Control Statements

1. **FizzBuzz**: Classic problem to practice conditional statements.
2. **Prime Number Check**: Determine if a number is prime using loops and conditionals.
3. **Palindrome Number**: Check if a number reads the same backward as forward.
4. **Armstrong Number**: Check if a number is an Armstrong number.
5. **Reverse Digits**: Reverse the digits of a number using loops.
6. **Factorial Calculation**: Compute the factorial of a number.
7. **Number of 1 Bits**: Count the number of 1s in the binary representation of a number.
8. **GCD and LCM Calculation**: Calculate the GCD and LCM of two numbers using loops.
9. **Find the nth Fibonacci Number**: Using both iterative and recursive approaches.
10. **Leap Year Check**: Write a program to check if a year is a leap year.

2. Recursion

1. **Fibonacci Sequence**: Find the nth Fibonacci number using recursion.
2. **Factorial of a Number**: Compute factorial recursively.
3. **Power Calculation**: Calculate `x^n` using recursion.
4. **Sum of Digits**: Recursively find the sum of digits of a number.
5. **Permutations of a String**: Generate all permutations of a string.
6. **Tower of Hanoi**: Solve the Tower of Hanoi problem.
7. **Subsets Generation**: Generate all subsets of a set (power set).
8. **Combination Sum**: Find all combinations that sum up to a target value.
9. **Palindrome Partitioning**: Partition a string into palindromes using recursion.
10. **Merge Sort**: Implement merge sort using recursion.

3. Arrays

1. **Two Sum**: Find two numbers in an array that add up to a target value.
2. **Best Time to Buy and Sell Stock**: Find the maximum profit you can achieve.
3. **Move Zeroes**: Move all zeroes to the end while maintaining the order of non-zero elements.
4. **Rotate Array**: Rotate the array to the right by k steps.
5. **Kadane's Algorithm**: Find the maximum sum subarray.
6. **Merge Intervals**: Merge overlapping intervals.
7. **Product of Array Except Self**: Calculate the product of all elements except the current one without division.
8. **Find the Duplicate Number**: Find the duplicate number in an array where elements are between 1 and n.
9. **Set Matrix Zeroes**: Modify the matrix such that if an element is 0, its entire row and column are set to 0.
10. **Find Missing Number**: Find the missing number in an array containing numbers from 1 to n.

4. String

1. **Reverse String**: Reverse a given string.
2. **Longest Palindromic Substring**: Find the longest palindromic substring.
3. **Valid Parentheses**: Check if the parentheses are balanced.
4. **String Compression**: Compress a string using the counts of repeated characters.
5. **Longest Substring Without Repeating Characters**: Find the length of the longest substring without repeating characters.
6. **Anagram Check**: Check if two strings are anagrams of each other.
7. **Count and Say**: Generate the nth term in the count-and-say sequence.
8. **String to Integer (atoi)**: Implement the function `` `atoi` `` which converts a string to an integer.
9. **Group Anagrams**: Group anagrams together from a list of strings.
10. **Minimum Window Substring**: Find the minimum window substring that contains all characters of another string.

5. Linked List

1. **Reverse Linked List**: Reverse a singly linked list.
2. **Detect Cycle in Linked List**: Check if a linked list has a cycle (using Floyd's cycle-finding algorithm).
3. **Merge Two Sorted Lists**: Merge two sorted linked lists.
4. **Remove Nth Node from End**: Remove the nth node from the end of the list.
5. **Linked List Cycle II**: Find the node where the cycle begins in a linked list.
6. **Palindrome Linked List**: Check if a linked list is a palindrome.
7. **Intersection of Two Linked Lists**: Find the intersection node of two linked lists.
8. **Remove Duplicates from Sorted List**: Remove duplicates from a sorted linked list.
9. **Add Two Numbers**: Add two numbers represented as linked lists.
10. **Flatten a Multilevel Doubly Linked List**: Flatten a multilevel doubly linked list.

6. Stack and Queue

Stack Problems:

1. **Valid Parentheses**: Check if the parentheses in an expression are balanced.
2. **Min Stack**: Design a stack that supports push, pop, top, and retrieving the minimum element i constant time.
3. **Evaluate Reverse Polish Notation**: Evaluate the value of an arithmetic expression in Reverse Polish Notation.
4. **Daily Temperatures**: Find the number of days you have to wait until a warmer temperature.
5. **Next Greater Element I**: Find the next greater element for each element of an array.

Queue Problems:

1. **Implement Queue using Stacks**: Implement a queue using two stacks.
2. **Circular Queue Implementation**: Implement a circular queue.
3. **Sliding Window Maximum**: Find the maximum value in every window of size k in an array.
4. **First Unique Character in a String**: Find the first non-repeating character in a string.
5. **Rotting Oranges**: Determine the time required to rot all oranges in a grid.

Control Statements

1. **Reverse Integer** (LeetCode 7)
2. **Roman to Integer** (LeetCode 13)
3. **Integer to Roman** (LeetCode 12)
4. **Valid Parentheses** (LeetCode 20)
5. **Merge Two Sorted Lists** (LeetCode 21)
6. **Remove Duplicates from Sorted Array** (LeetCode 26)
7. **Remove Element** (LeetCode 27)
8. **Find the Duplicate Number** (LeetCode 287)
9. **Longest Common Prefix** (LeetCode 14)
10. **Missing Number** (LeetCode 268)

Recursion

1. **Combination Sum** (LeetCode 39)
2. **Permutations** (LeetCode 46)
3. **Subsets** (LeetCode 78)
4. **Word Search** (LeetCode 79)
5. **Generate Parentheses** (LeetCode 22)
6. **Letter Combinations of a Phone Number** (LeetCode 17)
7. **N-Queens** (LeetCode 51)
8. **Climbing Stairs** (LeetCode 70)
9. **Unique Paths III** (LeetCode 980)
10. **Palindrome Partitioning** (LeetCode 131)

Arrays

1. **Two Sum** (LeetCode 1)
2. **3Sum** (LeetCode 15)
3. **Container With Most Water** (LeetCode 11)
4. **Product of Array Except Self** (LeetCode 238)
5. **Maximum Subarray** (LeetCode 53)
6. **Find Minimum in Rotated Sorted Array** (LeetCode 153)
7. **Search in Rotated Sorted Array** (LeetCode 33)
8. **Longest Consecutive Sequence** (LeetCode 128)
9. **Merge Intervals** (LeetCode 56)
10. **Next Permutation** (LeetCode 31)

Strings

1. **Longest Substring Without Repeating Characters** (LeetCode 3)
2. **Longest Palindromic Substring** (LeetCode 5)
3. **Zigzag Conversion** (LeetCode 6)
4. **String to Integer (atoi)** (LeetCode 8)
5. **Group Anagrams** (LeetCode 49)
6. **Valid Anagram** (LeetCode 242)
7. **Implement strStr()** (LeetCode 28)
8. **Longest Common Subsequence** (LeetCode 1143)
9. **Longest Repeating Substring** (LeetCode 1062)
10. **Minimum Window Substring** (LeetCode 76)

Linked Lists

1. **Reverse Linked List** (LeetCode 206)
2. **Remove Nth Node From End of List** (LeetCode 19)
3. **Linked List Cycle** (LeetCode 141)
4. **Merge Two Sorted Lists** (LeetCode 21)
5. **Merge k Sorted Lists** (LeetCode 23)
6. **Add Two Numbers** (LeetCode 2)
7. **Intersection of Two Linked Lists** (LeetCode 160)
8. **Reorder List** (LeetCode 143)
9. **Copy List with Random Pointer** (LeetCode 138)
10. **Flatten a Multilevel Doubly Linked List** (LeetCode 430)

Stacks and Queues

1. **Evaluate Reverse Polish Notation** (LeetCode 150)
2. **Simplify Path** (LeetCode 71)
3. **Daily Temperatures** (LeetCode 739)
4. **Next Greater Element I** (LeetCode 496)
5. **Min Stack** (LeetCode 155)
6. **Implement Queue using Stacks** (LeetCode 232)
7. **Decode String** (LeetCode 394)
8. **Design Circular Queue** (LeetCode 622)
9. **Sliding Window Maximum** (LeetCode 239)
10. **Largest Rectangle in Histogram** (LeetCode 84)

Trees

1. **Binary Tree Inorder Traversal** (LeetCode 94)
2. **Maximum Depth of Binary Tree** (LeetCode 104)
3. **Same Tree** (LeetCode 100)
4. **Invert Binary Tree** (LeetCode 226)
5. **Symmetric Tree** (LeetCode 101)
6. **Binary Tree Level Order Traversal** (LeetCode 102)
7. **Lowest Common Ancestor of a Binary Search Tree** (LeetCode 235)
8. **Balanced Binary Tree** (LeetCode 110)
9. **Binary Tree Maximum Path Sum** (LeetCode 124)
10. **Construct Binary Tree from Preorder and Inorder Traversal** (LeetCode 105)

Graphs

1. **Number of Islands** (LeetCode 200)
2. **Course Schedule** (LeetCode 207)
3. **Clone Graph** (LeetCode 133)
4. **Pacific Atlantic Water Flow** (LeetCode 417)
5. **Word Ladder** (LeetCode 127)
6. **Connected Components in an Undirected Graph** (LeetCode 323)
7. **Detect Cycle in a Directed Graph** (LeetCode 207 - Course Schedule can be used for this)
8. **Rotting Oranges** (LeetCode 994)
9. **Alien Dictionary** (LeetCode 269 - Similar problems exist)
10. **Graph Valid Tree** (LeetCode 261)

Advanced Tree Problems

1. **Flatten Binary Tree to Linked List** (LeetCode 114)
2. **Serialize and Deserialize Binary Tree** (LeetCode 297)
3. **Binary Tree Right Side View** (LeetCode 199)
4. **Kth Smallest Element in a BST** (LeetCode 230)
5. **Binary Search Tree Iterator** (LeetCode 173)
6. **Recover Binary Search Tree** (LeetCode 99)
7. **Populating Next Right Pointers in Each Node** (LeetCode 116)
8. **Construct Binary Tree from Inorder and Postorder Traversal** (LeetCode 106)
9. **House Robber III** (LeetCode 337)
10. **Unique Binary Search Trees** (LeetCode 96)

Advanced Graph Problems

1. **Find Eventual Safe States** (LeetCode 802)
2. **Redundant Connection** (LeetCode 684)
3. **Minimum Cost to Connect All Points** (LeetCode 1584)
4. **Critical Connections in a Network** (LeetCode 1192)
5. **Cheapest Flights Within K Stops** (LeetCode 787)
6. **Reconstruct Itinerary** (LeetCode 332)
7. **Longest Increasing Path in a Matrix** (LeetCode 329)
8. **Course Schedule II** (LeetCode 210)
9. **Graph Coloring** (LeetCode - Similar problem: 847. Shortest Path Visiting All Nodes)
10. **Network Delay Time** (LeetCode 743)