COMPETITIVE PROGRAMMING - CHALLENGES

1. Basic Data Structures & Input/Output

- **Objective**: Get familiar with fundamental data structures and how to handle input/output efficiently.
- **Topics Covered**: Arrays, strings, lists, stacks, queues.
- **Tasks**:
    - Learn how to read input and output data quickly (important for time-sensitive problems).
    - Solve problems like reversing a string, finding the maximum element in an array, and simple array manipulations.
- **Example Challenges**:
    - Reverse a string or array.
    - Find the sum of elements in an array.
    - Check if a string is a palindrome.

2. Basic Algorithms

- **Objective**: Understand basic algorithms and apply them to solve problems.
- **Topics Covered**: Sorting, searching, simple mathematics (e.g., finding GCD, LCM), and basic recursion.
- **Tasks**:
    - Implement and use common algorithms like binary search and bubble sort.
    - Practice recursive solutions for problems like factorial or Fibonacci numbers.

- **Example Challenges**:
    - Find the first occurrence of an element in a sorted array using binary search.
    - Sort an array of numbers using bubble or selection sort.
    - Calculate the GCD or LCM of two numbers.

3. Greedy Algorithms & Simple Dynamic Programming

- **Objective**: Begin working with greedy algorithms and get a basic understanding of dynamic programming (DP).
- **Topics Covered**: Greedy methods (e.g., coin change problem), introduction to DP with problems like Fibonacci.
- **Tasks**:
    - Solve problems using greedy techniques where local optimization leads to a global solution.
    - Understand and implement simple DP solutions.
- **Example Challenges**:
    - Minimum coins to make a value.
    - Find the nth Fibonacci number using dynamic programming.
    - Maximum sum of non-adjacent elements in an array.

4. Basic Graph Theory & Tree Problems

- **Objective**: Get introduced to basic graph and tree structures and algorithms.
- **Topics Covered**: Depth-First Search (DFS), Breadth-First Search (BFS), basic tree traversal (in-order, pre-order, post-order).
- **Tasks**:

    - Solve problems that involve navigating through graphs or trees, finding shortest paths, or traversing nodes.
- **Example Challenges**:
    - Implement BFS or DFS to traverse a graph.
    - Find the shortest path in an unweighted graph.
    - Calculate the height of a binary tree.