# PG-DAC Placement Preparation: Arrays, Strings & Numbers

## ARRAYS - Basic Level

1. **Find the maximum/minimum element -** Iterate and track max/min
2. **Calculate array sum and average -** Basic iteration and accumulation
3. **Reverse an array -** Using two pointers or recursion
4. **Check if array is sorted -** Single pass comparison
5. **Find second largest/smallest element -** Track top 2 elements
6. **Count occurrences of element -** Iterate and count
7. **Remove duplicates from sorted array -** Two pointer approach
8. **Merge two sorted arrays -** Two pointer merge technique
9. **Rotate array by k positions -** Left/right rotation
10. **Linear search vs Binary search -** Implement both

## ARRAYS - Next Level

1. **Two Sum Problem -** Hash map approach, O(n) time
2. **Container With Most Water -** Two pointer greedy approach
3. **Trapping Rain Water -** Dynamic programming or stack
4. **Maximum Subarray (Kadane's Algorithm) -** Contiguous subarray sum
5. **Product of Array Except Self -** Prefix and suffix products
6. **First Missing Positive -** Array mapping technique
7. **Median of Two Sorted Arrays -** Binary search approach
8. **Next Permutation -** In-place manipulation
9. **3Sum Problem -** Sorting + two pointer
10. **Set Matrix Zeroes -** In-place modification
11. **Search in Rotated Sorted Array -** Modified binary search
12. **Jump Game -** Greedy or dynamic programming

---

## STRINGS - Basic Level

1. **Reverse a string -** Two pointer or recursion
2. **Check palindrome -** Compare characters from both ends

3. **Count vowels and consonants -** Iterate through string
4. **String concatenation -** Join multiple strings
5. **Find character frequency -** Hash map or array counter
6. **Convert case (uppercase/lowercase) -** Built-in functions understanding
7. **Remove whitespaces -** Filter spaces
8. **Check if strings are anagram -** Sort and compare or frequency map
9. **String comparison and equality -** Basic comparison operations
10. **Find substring -** indexOf or manual search

## STRINGS - Next Level

1. **Longest Substring Without Repeating Characters -** Sliding window with hash map
2. **Longest Palindromic Substring -** Expand around center or DP
3. **Regular Expression Matching -** Dynamic programming
4. **Palindrome Number/String -** Two pointer approach
5. **Reverse Words in String -** Stack or reverse then reverse words
6. **Group Anagrams -** Hash map grouping by sorted string
7. **Minimum Window Substring -** Sliding window technique
8. **Word Ladder -** BFS approach
9. **Edit Distance (Levenshtein) -** Dynamic programming
10. **Valid Parentheses -** Stack-based solution
11. **ZigZag Conversion -** Pattern recognition
12. **Longest Common Prefix -** Character-by-character comparison

---

## NUMBERS - Basic Level

1. **Check if number is even/odd -** Modulo operation
2. **Find digits of a number -** Repeated division by 10
3. **Sum of digits -** Extract and add digits
4. **Reverse a number -** Build reversed number digit by digit
5. **Check prime number -** Divisibility check up to $\sqrt{n}$
6. **Factorial calculation -** Iterative or recursive
7. **Check perfect square -** Square root comparison

8. **Fibonacci sequence -** Iterative or recursive generation
9. **GCD (Greatest Common Divisor) -** Euclidean algorithm
10. **LCM (Least Common Multiple) -** Using GCD formula

### NUMBERS - Next Level

1. **Power(x, n) -** Binary exponentiation, handle negative powers
2. **Integer to Roman/Roman to Integer -** Mapping and logic
3. **Palindrome Number -** Without converting to string
4. **Happy Number -** Cycle detection
5. **Ugly Number -** Prime factor approach
6. **Missing Number in Array -** XOR or sum approach
7. **Majority Element -** Boyer-Moore voting algorithm
8. **Single Number (XOR) -** Bit manipulation
9. **Fraction to Recurring Decimal -** Hash map for cycle detection
10. **Excel Column Number/Title Conversion -** Base conversion logic
11. **Factorial Trailing Zeroes -** Count factors of 5
12. **Count Primes (Sieve of Eratosthenes) -** Efficient prime generation

---

### Study Tips

**Basic Level Mastery:**

- Focus on understanding time and space complexity
- Practice manual tracing with small examples
- Master two-pointer technique
- Get comfortable with hash maps/dictionaries

**Next Level Approach:**

- Recognize problem patterns (sliding window, two pointer, DP, etc.)
- Think about optimization from brute force
- Practice on platforms like LeetCode, HackerRank
- Solve 2-3 variations of same pattern

**Time Management:**

- Spend 60% time on basic concepts
- Allocate 40% time on next level problems
- Practice 5-7 problems daily
- Review solutions and optimize code

**Important for PG-DAC:**

- Code without IDE - practice on paper/whiteboard
- Explain approach before coding
- Test with edge cases (empty, single element, duplicates)
- Mention time and space complexity