**In C**

**1. Sum of Digits**

* **Problem**: Write a C program to calculate the sum of the digits of an integer.
* **Input**: A single integer (e.g., 1234).
* **Output**: The sum of its digits (e.g., 1 + 2 + 3 + 4 = 10).
* **Enhances**: Loops, conditionals, modulus operator, basic arithmetic.

**2. Palindrome Checker**

* **Problem**: Write a C program that checks if a given string or integer is a palindrome (reads the same backward as forward).
* **Input**: A string or integer (e.g., "madam" or 121).
* **Output**: Yes or No.
* **Enhances**: Strings, arrays, loops, conditionals.

**3. Prime Number Finder**

* **Problem**: Write a program to check whether a number is prime or not, and find all prime numbers up to a given number n.
* **Input**: A single integer n.
* **Output**: All prime numbers less than or equal to n.
* **Enhances**: Loops, conditionals, efficiency considerations.

**4. Caesar Cipher**

* **Problem**: Implement a Caesar cipher where the user inputs a string and a shift value, and the program outputs the encrypted message.
* **Input**: A string and a shift integer.
* **Output**: Encrypted message using the Caesar cipher.
* **Enhances**: Arrays, strings, ASCII manipulation.

**5. Factorial Calculation**

* **Problem**: Write a program to calculate the factorial of a number using both an iterative approach and a recursive approach.
* **Input**: A single integer.
* **Output**: The factorial of that number.
* **Enhances**: Recursion, loops, conditionals.

**6. Fibonacci Sequence**

* **Problem**: Write a program to generate the first n Fibonacci numbers.
* **Input**: A single integer n (number of terms).
* **Output**: The first n numbers in the Fibonacci sequence.
* **Enhances**: Loops, conditionals, recursive thinking.

**7. Matrix Multiplication**

* **Problem**: Write a C program to multiply two matrices.
* **Input**: Two 2D matrices of compatible dimensions.
* **Output**: The product of the two matrices.
* **Enhances**: Arrays, nested loops, mathematics.

**8. Reverse a String**

* **Problem**: Write a function to reverse a given string without using the standard library function strrev().
* **Input**: A string.
* **Output**: The reversed string.
* **Enhances**: Pointers, string manipulation, loops.

**9. Merge Two Sorted Arrays**

* **Problem**: Write a C program that merges two sorted arrays into a single sorted array.
* **Input**: Two sorted arrays.
* **Output**: A new sorted array that contains elements of both input arrays.
* **Enhances**: Arrays, loops, sorting logic.

**10. Binary Search**

* **Problem**: Implement a binary search algorithm to search for a number in a sorted array.
* **Input**: A sorted array and the target value.
* **Output**: The index of the target value or -1 if not found.
* **Enhances**: Arrays, divide-and-conquer algorithms.

**11. Pascal's Triangle**

* **Problem**: Write a C program to print the first n rows of Pascal's triangle.
* **Input**: A single integer n.
* **Output**: The first n rows of Pascal's triangle.
* **Enhances**: Loops, nested loops, mathematics (combinatorics).

**12. Unique Elements in an Array**

* **Problem**: Write a C program that finds the unique elements in an array.
* **Input**: An array of integers with possibly duplicate elements.
* **Output**: A new array containing only unique elements.
* **Enhances**: Arrays, loops, conditionals.

**13. Towers of Hanoi**

* **Problem**: Implement the Towers of Hanoi problem, where the user inputs the number of disks, and the program prints the steps to move the disks.
* **Input**: Number of disks.
* **Output**: Steps to solve the puzzle.
* **Enhances**: Recursion, problem decomposition.

**14. Text File I/O: Word Count**

* **Problem**: Write a C program to read a text file and count the number of words, lines, and characters.
* **Input**: A text file.
* **Output**: Number of words, lines, and characters in the file.
* **Enhances**: File I/O, string manipulation.

**15. Sudoku Solver**

* **Problem**: Write a C program that solves a given Sudoku puzzle using backtracking.
* **Input**: A 9x9 grid representing the Sudoku puzzle.
* **Output**: The solved puzzle.
* **Enhances**: Recursion, backtracking, arrays.

**16. Decimal to Binary Converter**

* **Problem**: Write a program to convert a given decimal number into its binary equivalent.
* **Input**: A decimal number.
* **Output**: The binary representation of the number.
* **Enhances**: Loops, bitwise operations, mathematics.

**17. GCD and LCM**

* **Problem**: Write a program to find the Greatest Common Divisor (GCD) and Least Common Multiple (LCM) of two numbers.
* **Input**: Two integers.
* **Output**: The GCD and LCM of the two numbers.
* **Enhances**: Loops, conditionals, Euclidean algorithm.

**18. Circular Queue Implementation**

* **Problem**: Implement a circular queue in C using arrays.
* **Input**: Operations like enqueue, dequeue, and displaying the queue.
* **Output**: Perform the specified operations on the circular queue.
* **Enhances**: Data structures, arrays, modular arithmetic.

**19. Check Armstrong Number**

* **Problem**: Write a program to check whether a given number is an Armstrong number (a number equal to the sum of its own digits each raised to the power of the number of digits).
* **Input**: A single integer.
* **Output**: Yes or No (whether the number is an Armstrong number).
* **Enhances**: Loops, mathematical calculations, conditionals.

**20. Check Leap Year**

* **Problem**: Write a program to check whether a given year is a leap year.
* **Input**: A single year.
* **Output**: Yes or No (whether the year is a leap year).
* **Enhances**: Conditionals, modulus operator.

**21. Merge Sort**

* **Problem**: Implement the merge sort algorithm to sort an array of integers.
* **Input**: An array of integers.
* **Output**: The sorted array.
* **Enhances**: Divide and conquer, recursion, arrays.

**22. Tower Defense Simulation**

* **Problem**: Simulate a simple tower defense game where a series of "enemies" (with health points) march across the screen, and a "tower" reduces their health with each attack. When their health reaches zero, they are removed from the screen.
* **Input**: The number of enemies and their initial health values.
* **Output**: Track the health of the enemies after each tower attack until all enemies are defeated.
* **Enhances**: Loops, conditionals, arrays.

**23. Bank Account System**

* **Problem**: Create a bank account management system where users can deposit, withdraw, and check their balance.
* **Input**: Operations such as deposit, withdraw, and balance check.
* **Output**: The updated balance after each operation.
* **Enhances**: Functions, conditionals, input/output handling.

**24. String Permutations**

* **Problem**: Write a program to print all permutations of a given string.
* **Input**: A string.
* **Output**: All possible permutations of the string.
* **Enhances**: Recursion, string manipulation, backtracking.

**25. Roman Numeral Converter**

* **Problem**: Write a program that converts a given integer into its Roman numeral equivalent.
* **Input**: A single integer.
* **Output**: The Roman numeral equivalent.
* **Enhances**: Conditionals, arrays, loops, string handling.

**26. Sorting Student Records**

* **Problem**: Write a program to manage student records (name, age, grades). The program should allow you to add new records and sort them by name, age, or grades.
* **Input**: Student details and the field by which to sort.
* **Output**: Sorted list of students based on the chosen field.
* **Enhances**: Structs, arrays, sorting, conditionals.

**27. Count Vowels and Consonants**

* **Problem**: Write a program to count the number of vowels and consonants in a given string.
* **Input**: A string.
* **Output**: The number of vowels and consonants in the string.
* **Enhances**: Strings, conditionals, loops.

**28. File Copying Program**

* **Problem**: Write a program to copy the contents of one file to another file.
* **Input**: Two filenames (source and destination).
* **Output**: A copy of the source file in the destination file.
* **Enhances**: File handling, I/O operations.

**29. Temperature Converter**

* **Problem**: Write a program to convert temperatures between Fahrenheit, Celsius, and Kelvin.
* **Input**: A temperature and the units (F, C, K).
* **Output**: The temperature in the other two units.
* **Enhances**: Functions, mathematical operations, conditionals.

**30. Matrix Rotation**

* **Problem**: Write a program to rotate a 2D matrix (NxN) 90 degrees clockwise.
* **Input**: A square matrix.
* **Output**: The matrix rotated 90 degrees.
* **Enhances**: Arrays, loops, matrix manipulation.

**31. Find Missing Number in Array**

* **Problem**: Write a program to find the missing number in an array containing n unique numbers from 1 to n, except for one missing number.
* **Input**: An array of size n-1.
* **Output**: The missing number.
* **Enhances**: Arrays, loops, mathematical logic.

**32. Find Duplicates in an Array**

* **Problem**: Write a program to find and print all duplicate elements in an array.
* **Input**: An array of integers.
* **Output**: All duplicate elements in the array.
* **Enhances**: Arrays, loops, conditionals.

**33. Simple Calculator**

* **Problem**: Implement a simple calculator that performs basic arithmetic operations (addition, subtraction, multiplication, division) based on user input.
* **Input**: Two numbers and an operator.
* **Output**: The result of the operation.
* **Enhances**: Conditionals, functions, input/output handling.

**34. Find Maximum Subarray Sum**

* **Problem**: Write a program to find the contiguous subarray with the maximum sum in a given array (Kadane's Algorithm).
* **Input**: An array of integers.
* **Output**: The maximum sum of the subarray.
* **Enhances**: Arrays, dynamic programming, optimization.

**35. Implement a Stack Using Arrays**

* **Problem**: Implement a stack data structure using arrays with push, pop, and display operations.
* **Input**: Operations to push or pop elements.
* **Output**: The result of the stack operations.
* **Enhances**: Data structures, arrays, conditionals, stack implementation.