Beginner Level

1. Write a C program to print the ASCII values of all characters.

2. Calculate the area of a rectangle using a function that takes the length and width as inputs.

3. Write a program to find the greatest common divisor (GCD) of two numbers using the Euclidean algorithm.

4. Implement a simple calculator program that can perform addition, subtraction, multiplication, and division.

5. Create a program to check if a given year is a leap year.

6. Generate a pattern of right-angled triangles using nested loops.

7. Implement a program that converts temperature between Celsius and Fahrenheit.

8. Write a program to find the smallest element in an array.

9. Calculate the sum of digits of a number using recursion.

10. Create a program to reverse a string without using any library functions.

Intermediate Level

11. Implement a stack data structure with push, pop, and peek operations.

12. Write a program to check if a string is an anagram of another string.

13. Develop a program that performs binary search on a sorted array.

14. Implement a queue using two stacks.

15. Create a program that counts the occurrences of a word in a given text file.

16. Write a program to find the intersection of two arrays.

17. Implement a circular linked list with insertion, deletion, and traversal operations.

18. Develop a program to convert a decimal number to its hexadecimal representation.

19. Write a function to reverse a linked list in groups of a specified size.

20. Implement a basic text-based game using C programming.

Advanced Level

21. Write a program to implement depth-first search (DFS) on a graph.

22. Implement a memory allocator (malloc and free functions) using a linked list of free blocks.

23. Develop a program that simulates a basic shell with history and command execution features.

24. Implement a priority queue using a binary heap.

25. Create a program to parse and evaluate simple mathematical expressions.

26. Write a program to perform matrix inversion using the Gauss-Jordan elimination method.

27. Implement a basic garbage collector for managing dynamic memory allocation.

28. Develop a program that simulates a multi-threaded file downloader.

29. Write a program to solve the N-Queens problem using backtracking.

30. Implement a hash map data structure with collision resolution techniques.

Expert Level

31. Create a program that simulates a basic operating system process scheduler.

32. Implement a concurrent server that handles multiple client connections using threads or processes.

33. Write a program to perform image compression using a specified algorithm (e.g., Huffman coding).

34. Develop a program that performs audio signal processing, such as filtering or Fourier transformation.

35. Implement a distributed computing application using network sockets and a master-worker architecture.

36. Write a program to perform sentiment analysis on a text document using machine learning techniques.

37. Develop a program that uses parallel computing to solve a complex scientific computation problem.

38. Implement a basic compiler for a simple programming language using lexing, parsing, and code generation.

39. Create a program that performs real-time video processing and applies filters to the video stream.

40. Write a program to solve a complex mathematical problem, such as prime number factorization, using parallel algorithms.