# Programming Problems in C language

Basic Concepts:

1. Write a program to find the size of different data types.
2. Explain the difference between call by value and call by reference with examples.
3. Implement a macro to find the maximum of two numbers.
4. Discuss the use of volatile keyword in C.
5. Explain the concept of a function pointer and provide an example.
6. Write a program to swap two variables without using a temporary variable.
7. Implement a circular linked list and perform basic operations.
8. Describe the process of memory allocation for a C program.
9. Discuss the role of the const keyword in C.

Advanced Pointers and Memory:

1. Explain the purpose of the restrict keyword in C.
2. Write a program to dynamically allocate a two-dimensional array.
3. Implement a function to concatenate two strings without using library functions.
4. Discuss the differences between malloc, calloc, and realloc.
5. Write a program to reverse a linked list in groups of given size.
6. Explain the concept of pointer arithmetic and demonstrate its use.

Advanced Data Structures:

1. Implement a trie data structure and perform insert and search operations.
2. Discuss the implementation and advantages of a skip list.
3. Write a program to detect a cycle in a directed graph.
4. Implement a priority queue using a binary heap.
5. Explain the concept of a self-adjusting list and its applications.

File Handling:

1. Write a program to copy the contents of one file to another.
2. Implement a file compression and decompression algorithm.
3. Discuss the use of fseek and ftell functions in file handling.
4. Write a program to serialize and deserialize a binary tree.

Advanced Algorithms:

1. Implement Dijkstra's algorithm for finding the shortest path in a graph.
2. Discuss the advantages and disadvantages of various graph traversal algorithms.
3. Write a program to find the longest common subsequence of two strings.
4. Implement the A\* algorithm for pathfinding in a grid.
5. Discuss the concept of topological sorting and its applications.

Network Programming:

1. Write a simple TCP server-client program in C.
2. Implement a basic DNS resolver in C.
3. Discuss the use of raw sockets in network programming.
4. Write a program to perform a simple HTTP server.

Multi-threading and Synchronization:

1. Implement a multi-threaded program using pthreads to calculate the factorial of a number.
2. Discuss the concept of thread synchronization and its importance.
3. Write a program to solve the producer-consumer problem using semaphores.
4. Implement a lock-free data structure using atomic operations.

Bit Manipulation:

1. Write a program to count the number of set bits in an integer.
2. Implement a function to find the single non-repeating element in an array.
3. Discuss the bitwise manipulation techniques to optimize code.

Dynamic Programming:

1. Solve the Knapsack problem using dynamic programming.
2. Implement the Floyd-Warshall algorithm for all-pairs shortest paths.
3. Write a program to find the edit distance between two strings.

System Programming:

1. Explain the role of system calls in operating systems.
2. Write a program to create a daemon process in C.
3. Implement a simple shell in C.

Cryptography:

1. Write a program to implement the Caesar cipher.
2. Implement the RSA algorithm for public-key cryptography.
3. Discuss the concept of digital signatures and implement one in C.

Database Connectivity:

1. Write a program to connect to a MySQL database in C.
2. Discuss the use of prepared statements in database programming.

Advanced String Manipulation:

1. Implement a program to check if two strings are anagrams.
2. Write a function to reverse words in a given sentence without using library functions.
3. Discuss the concept of string matching algorithms and implement one.

Error Handling:

1. Explain the use of setjmp and longjmp functions for error handling.
2. Implement a custom memory allocator with error checking.

Operating System Concepts:

1. Discuss the concept of virtual memory and its implementation.
2. Write a program to create and manage threads using the pthread library.

Floating Point Arithmetic:

1. Explain the limitations and challenges of floating-point arithmetic.
2. Write a program to perform numerical calculations with precision.

Real-time Systems:

1. Discuss the principles of real-time systems.
2. Implement a real-time scheduling algorithm.

Interprocess Communication:

1. Write a program to demonstrate interprocess communication using pipes.
2. Implement a shared memory mechanism for interprocess communication.

Advanced Macros:

1. Create a macro to calculate the factorial of a number.
2. Discuss the use of token pasting and stringizing operators in macros.

Advanced Preprocessor Directives:

1. Implement a program using conditional compilation directives.
2. Discuss the concept of include guards and their importance.

Secure Coding:

1. Explain the principles of secure coding in C.
2. Write a program to prevent buffer overflow vulnerabilities.

Web Development in C:

1. Discuss the use of CGI in web development using C.
2. Implement a basic web server in C.

Artificial Intelligence in C:

1. Write a program to implement a basic neural network in C.
2. Discuss the use of genetic algorithms in optimization problems.

Game Development:

1. Implement a simple game using the SDL library in C.
2. Discuss the principles of game loop implementation.

Code Optimization:

1. Write an optimized version of a sorting algorithm in C.
2. Discuss the techniques for code profiling and optimization.

Compiler Design:

1. Explain the phases of a compiler and their functions.
2. Write a simple lexical analyzer using C.

Embedded Systems:

1. Implement a program to interface with GPIO pins on a microcontroller.
2. Discuss the role of interrupts in embedded systems.

Parallel Programming:

1. Write a program to perform parallel matrix multiplication using OpenMP.
2. Discuss the principles of parallel programming and its challenges.

Internet of Things (IoT):

1. Implement a simple IoT application using C.
2. Discuss the challenges and security considerations in IoT programming.

Robotics:

1. Write a program to control a robot using C.
2. Discuss the principles of robot programming and control.

Machine Learning in C:

1. Implement a basic machine learning algorithm in C.
2. Discuss the challenges of implementing machine learning in resource-constrained environments.

GPU Programming:

1. Write a program to perform parallel computations using GPU in C.
2. Discuss the principles of GPU programming and CUDA.

Functional Programming in C:

1. Implement a simple functional programming concept in C.
2. Discuss the advantages and disadvantages of functional programming.

Signal Processing:

1. Write a program to perform basic signal processing operations.
2. Implement a simple audio processing application in C.

Advanced Graphics Programming:

1. Implement a basic graphics rendering program using OpenGL in C.
2. Discuss the principles of computer graphics programming.

Network Security:

1. Write a program to implement a basic firewall in C.
2. Discuss the principles of network security in C programming.