Question 1:
Implement an RPC system where the client can send complex data (e.g., arrays or structures) to the server for processing. For example:

- The server should provide a function to calculate the **average of an array of numbers** sent by the client.
- Use a serialization technique (e.g., converting the array into a string) to transfer data between the client and server.

Requirements:

- The client sends the array and size to the server.
- The server calculates the average and sends the result back to the client.

Question 2:
Write a program to simulate a Remote Procedure Call (RPC) system. Your system should include:

1. A **server** that hosts basic mathematical operations:
   - Addition (`add(a, b)`)
   - Subtraction (`subtract(a, b)`)
   - Multiplication (`multiply(a, b)`)
   - Division (`divide(a, b)`).
2. A **client** that sends a request to the server to execute one of these operations with two numbers and displays the result.

Requirements:

- Implement a simple protocol to encode the procedure name and parameters in the client request (e.g., `"add 5 3"`).
- The server should decode the request, execute the appropriate function, and send back the result.
- Handle errors like invalid procedure names and division by zero.

Question 3:
Create an RPC system in C to perform **file operations** on the server. The client should send requests to:

1. **Create** a file with specified content.
2. **Read** the contents of a file.
3. **Delete** a file.

Requirements:

- The client should specify the operation and file name (and content for file creation).
- The server should process the request and send back the result or an acknowledgment.

- Ensure proper error handling for non-existent files and other invalid operations.

Question 4:
Enhance the RPC system by adding **authentication**:

1. The client must send a username and password along with the request.
2. The server should validate the credentials before processing the request.
3. If authentication fails, the server should reject the request and notify the client.

Requirements:

- Store valid credentials in a file or in-memory on the server.
- Handle cases where the credentials are incorrect.

Question 5:
Create an RPC system where the server provides a **logging service**:

1. The client sends log messages (`INFO`, `WARN`, `ERROR`) with timestamps to the server.
2. The server saves the logs to a file and sends back an acknowledgment.

Requirements:

- Include an option for the client to request all logs or logs of a specific type (e.g., only `ERROR` logs).