Question 3

Conceptual Server Algorithm

1. **Initialization and Setup**
    - Create a UDP socket (sockfd).
    - Bind the socket to a specified port (PORT) on the server.
    - Print a message indicating the server is listening on the specified port.
2. **Handling Client Requests**
    - Enter an infinite loop to continuously listen for incoming client requests.
    - For each incoming request:
        - Allocate memory for client_info to store client socket information and choice.
        - Receive the client's choice of operation (1 for displaying catalog, 2 for searching a book, 3 for ordering a book, 4 for paying for a book).
        - Create a new thread to handle the client request using pthread_create() and pass the client_info structure to the thread.
3. **Thread Function (handle_client)**:
    - Receive the client_info structure containing client socket information and choice.
    - Based on the client's choice:
        - **Choice 1 (Display Catalog)**:
            - Read book information from bookfile.txt.
            - Send the count of books followed by each book's details to the client.
        - **Choice 2 (Search Book)**:
            - Receive search criteria (either title or ISBN) from the client.
            - Search for the book in bookfile.txt.
            - Send search status ("found" or "not found") and book details to the client.
        - **Choice 3 (Order Book)**:
            - Receive the ISBN of the book to be ordered from the client.
            - Search for the book in bookfile.txt.
            - If found, update the status to "ordered" in orders.txt, generate an order number, and send the order number to the client.
        - **Choice 4 (Pay for Book)**:
            - Receive the order number for payment from the client.
            - Search for the order in orders.txt.
            - If found, update the status to "paid" in orders.txt and confirm payment to the client.
4. **Cleanup**
    - Free allocated memory for client_info after handling each client request.
    - Close the socket (sockfd) when the server is terminated.

Question 3

Application Protocol

The application protocol describes how the server communicates with clients over UDP:

1. **Message Format**:
    - **Client to Server**:
        - **Request Message**: Sent when the client selects an operation.
            - Format: [operation_choice]
            - Example: 1 (for displaying catalog), 2 [search_type] [search_criteria] (for searching a book), etc.
        - **Data Transfer**: Additional messages as needed to transfer data such as book details or search results.
    - **Server to Client**:
        - **Response Message**: Sent in response to client requests.
            - Format: [status_code] [data]
            - Example: 200 OK [count_of_books] (for catalog display), 404 Not Found (for book not found during search), etc.
        - **Data Transfer**: Additional messages to send book details, order numbers, or payment status.
2. **Operations**:
    - **Display Catalog**:
        - **Client Request**: 1
        - **Server Response**: 200 OK [count_of_books], followed by book details.
    - **Search Book**:
        - **Client Request**: 2 [search_type] [search_criteria]
        - **Server Response**: 200 OK [book_details] or 404 Not Found if the book is not found.
    - **Order Book**:
        - **Client Request**: 3 [ISBN]
        - **Server Response**: 200 OK [order_number] if successfully ordered, 404 Not Found if book with given ISBN doesn't exist.
    - **Pay for Book**:
        - **Client Request**: 4 [order_number]
        - **Server Response**: 200 OK if payment is successful, 404 Not Found if order with given number doesn't exist.
3. **Error Handling**:
    - Use HTTP-like status codes (200 OK, 404 Not Found) to indicate success or failure of operations.
    - Handle errors such as file not found (500 Internal Server Error), socket errors, etc., with appropriate error messages.
4. **Concurrency**:
    - Use threads (pthread) to handle multiple client requests concurrently.
    - Ensure thread safety when accessing shared resources like file handles (bookfile.txt, orders.txt) by using file locks or synchronization mechanisms.
5. **Data Format**:
    - Use structured data formats (like CSV or tab-separated values) for storing book and order information in files (bookfile.txt, orders.txt).
    - Ensure consistent parsing and formatting of data between server and client to prevent data corruption or misinterpretation.