Exercise 3: TCP Client-Server Echo Program

Objective

Learn how to use **TCP sockets** in C++ to establish a connection and exchange data reliably.

Task

You will write two C++ programs:

1. Echo Server

- Listens on a TCP port (e.g., `5001`).
- Accepts client connections.
- Receives a message and sends the **same message back** (echo).

2. Echo Client

- Connects to the server.
- Sends a message (e.g., "Hello Server!").
- Receives the echoed message and prints it.

Optional Challenge

- Make the server handle multiple clients concurrently (using threads).
- Add simple error handling for disconnected clients.

Starter Code

echo_server.cpp (Stub)

```
#include <iostream>
#include <cstring>
#include <unistd.h>
#include <arpa/inet.h>

int main() {
    int server_fd, client_fd;
    struct sockaddr_in server_addr, client_addr;
    socklen_t addr_len = sizeof(client_addr);
```

```
    // TODO: Create TCP socket

    // TODO: Bind to a port

    // TODO: Listen for connections

    // TODO: Accept client connection

    // TODO: Receive message from client

    // TODO: Send the same message back (echo)

    // TODO: Close connections

    return 0;
}
```

echo_client.cpp (Stub)

```
#include <iostream>
#include <cstring>
#include <unistd.h>
#include <arpa/inet.h>

int main() {
    int sockfd;
    struct sockaddr_in server_addr;

    // TODO: Create TCP socket

    // TODO: Connect to server

    // TODO: Send message to server

    // TODO: Receive echo message

    // TODO: Close connection

    return 0;
}
```

Learning Goals

- Understand the difference between **UDP** and **TCP** communication.
- Practice establishing and maintaining a **connection-oriented** communication.
- Experience how **reliable data transmission** works in TCP.

Optional Challenge

- Add support for multiple concurrent client connections using **threads**.
- Implement simple connection error handling.

Suggested Questions for Students

1. How does TCP differ from UDP?
2. What steps are required to establish a TCP connection?
3. What happens if the client disconnects unexpectedly?
4. How can you make the server handle multiple clients?

Hint

You will need to look up these functions:

- `socket()`
- `bind()`
- `listen()`
- `accept()`
- `connect()`
- `send()`
- `recv()`
- `close()`