Practical No. 6

Code :

tcp_server.cpp :

```
#include <iostream>
#include <winsock2.h>
#include <ws2tcpip.h>
#include <fstream>
using namespace std;
#pragma comment(lib, "ws2_32.lib")
#define PORT 8080
#define BUFFER_SIZE 1024
int main() {
  WSADATA wsa;
  if (WSAStartup(MAKEWORD(2,2), &wsa) != 0) {
    cout << "WSAStartup failed" << endl;
    return 1;   }
  SOCKET serverSock, clientSock;
  struct sockaddr_in serverAddr, clientAddr;
  int clientSize = sizeof(clientAddr);
  serverSock = socket(AF_INET, SOCK_STREAM, 0);
  if (serverSock == INVALID_SOCKET) {
    cout << "Socket creation failed" << endl;
    WSACleanup();
    return 1;   }
  serverAddr.sin_family = AF_INET;
  serverAddr.sin_addr.s_addr = INADDR_ANY;
  serverAddr.sin_port = htons(PORT);
  if (bind(serverSock, (struct sockaddr*)&serverAddr, sizeof(serverAddr)) == SOCKET_ERROR) {
    cout << "Bind failed" << endl;
```

```
        closesocket(serverSock);
        WSACleanup();
        return 1;    }
    if (listen(serverSock, 1) == SOCKET_ERROR) {
        cout << "Listen failed" << endl;
        closesocket(serverSock);
        WSACleanup();
        return 1;  }
    cout << "Waiting for client..." << endl;
    clientSock = accept(serverSock, (struct sockaddr*)&clientAddr, &clientSize);
    if (clientSock == INVALID_SOCKET) {
        cout << "Accept failed" << endl;
        closesocket(serverSock);
        WSACleanup();
        return 1;}
    cout << "Client connected!" << endl;
// ===== Part A: Say Hello =====
    char buffer[BUFFER_SIZE];
    int bytesReceived = recv(clientSock, buffer, BUFFER_SIZE, 0);
    buffer[bytesReceived] = '\0';
    cout << "Client: " << buffer << endl;
    string helloMsg = "Hello from Server";
    send(clientSock, helloMsg.c_str(), helloMsg.length(), 0);
    // ===== Part B: File Transfer =====
    cout << "Receiving file from client..." << endl;
    ofstream outFile("received_file.txt", ios::binary);
    while ((bytesReceived = recv(clientSock, buffer, BUFFER_SIZE, 0)) > 0) {
        outFile.write(buffer, bytesReceived);
        if (bytesReceived < BUFFER_SIZE) break; // End of file    }
    outFile.close();
    cout << "File received successfully!" << endl;
    closesocket(clientSock);
    closesocket(serverSock);
    WSACleanup();
    return 0;}
```

Tcp_client.cpp :

```
#include <iostream>
#include <winsock2.h>
#include <ws2tcpip.h>
#include <fstream>
using namespace std;
#pragma comment(lib, "ws2_32.lib")
#define PORT 8080
#define BUFFER_SIZE 1024
int main() {
    WSADATA wsa;
    if (WSAStartup(MAKEWORD(2,2), &wsa) != 0) {
        cout << "WSAStartup failed" << endl;
        return 1;}
    SOCKET sock;
    struct sockaddr_in serverAddr;
    sock = socket(AF_INET, SOCK_STREAM, 0);
    if (sock == INVALID_SOCKET) {
        cout << "Socket creation failed" << endl;
        WSACleanup();
        return 1;}
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(PORT);
    serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
```

```
  if (connect(sock, (struct sockaddr*)&serverAddr, sizeof(serverAddr)) < 0) {
    cout << "Connection failed" << endl;
    closesocket(sock);
    WSACleanup();
    return 1;}
  cout << "Connected to server!" << endl;
  // ===== Part A: Say Hello =====
  string helloMsg = "Hello from Client";
  send(sock, helloMsg.c_str(), helloMsg.length(), 0);
  char buffer[BUFFER_SIZE];
  int bytesReceived = recv(sock, buffer, BUFFER_SIZE, 0);
  buffer[bytesReceived] = '\0';
  cout << "Server: " << buffer << endl;
  // ===== Part B: Send File =====
  ifstream inFile("file_to_send.txt", ios::binary);
  if (!inFile) {
    cout << "File not found!" << endl;
  } else {
    cout << "Sending file to server..." << endl;
    while (!inFile.eof()) {
      inFile.read(buffer, BUFFER_SIZE);
      send(sock, buffer, inFile.gcount(), 0);}
    inFile.close();
    cout << "File sent successfully!" << endl;    }
  closesocket(sock);
  WSACleanup();
  return 0;}
```

File_to_sent.txt :

Hello Server!

This is a test file sent from the TCP client.

It contains multiple lines to check if the file transfer works properly.

Line 1: TCP communication is working.

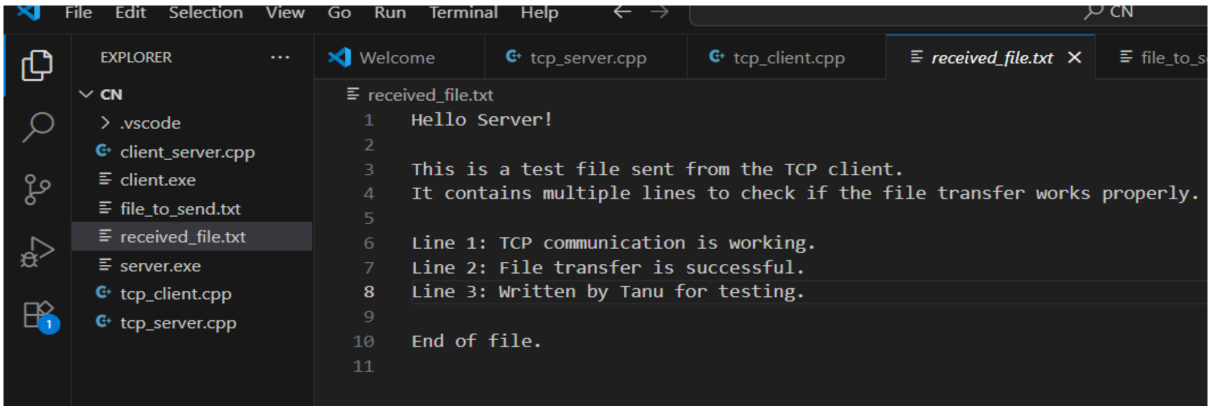
Line 2: File transfer is successful.

Line 3: Written by Tanu for testing.

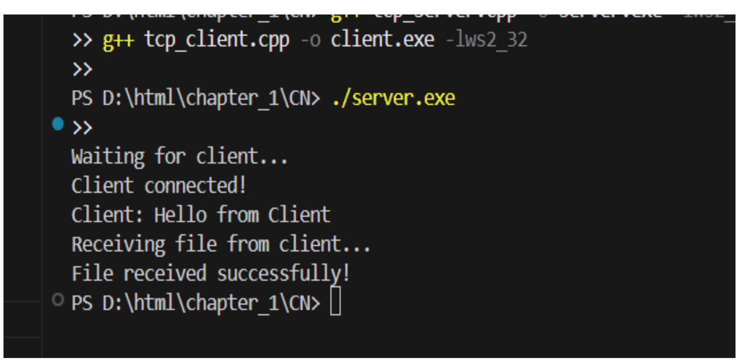
End of file.

Output :

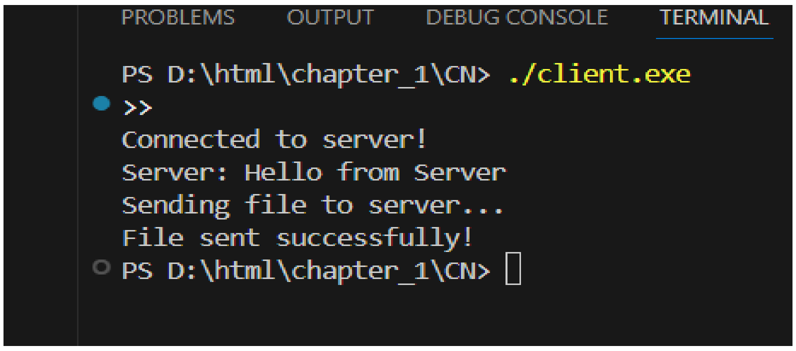
received_file.txt :



```
Hello Server!

This is a test file sent from the TCP client.
It contains multiple lines to check if the file transfer works properly.

Line 1: TCP communication is working.
Line 2: File transfer is successful.
Line 3: Written by Tanu for testing.

End of file.

```



```
>> g++ tcp_client.cpp -o client.exe -lws2_32
>>
PS D:\html\chapter_1\CN> ./server.exe
>>
Waiting for client...
Client connected!
Client: Hello from Client
Receiving file from client...
File received successfully!
PS D:\html\chapter_1\CN>
```



```
PS D:\html\chapter_1\CN> ./client.exe
>>
Connected to server!
Server: Hello from Server
Sending file to server...
File sent successfully!
PS D:\html\chapter_1\CN>
```