33. Implementing the applications using TCP file transfer in java/C.

Client

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <winsock2.h>

#define PORT 8080
#define BUFFER_SIZE 1024
#define OUTPUT_FILE "received.txt"

int main() {
    WSADATA wsa;
    SOCKET sock;
    struct sockaddr_in server_addr;
    FILE *file;
    char buffer[BUFFER_SIZE] = {0};

    // Initialize Winsock
    WSAStartup(MAKEWORD(2, 2), &wsa);

    // Create socket
    sock = socket(AF_INET, SOCK_STREAM, 0);
    if (sock == INVALID_SOCKET) {
        perror("Socket creation failed");
        WSACleanup();
        exit(EXIT_FAILURE);
    }

    // Server details
    server_addr.sin_family = AF_INET;
```

```
server_addr.sin_port = htons(PORT);
server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

// Connect to server
if (connect(sock, (struct sockaddr *)&server_addr, sizeof(server_addr)) == SOCKET_ERROR) {
    perror("Connection failed");
    closesocket(sock);
    WSACleanup();
    exit(EXIT_FAILURE);
}

printf("Connected to server. Receiving file...\n");

// Open file to save data
file = fopen(OUTPUT_FILE, "w");
if (file == NULL) {
    perror("Failed to create file");
    closesocket(sock);
    WSACleanup();
    exit(EXIT_FAILURE);
}

// Receive data
int bytes_received;
while ((bytes_received = recv(sock, buffer, BUFFER_SIZE, 0)) > 0) {
    fwrite(buffer, 1, bytes_received, file);
}

printf("File received successfully!\n");

// Cleanup
```

```
    fclose(file);
    closesocket(sock);
    WSACleanup();

    return 0;
}
```

Server:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <winsock2.h>
#include <ws2tcpip.h>

#pragma comment(lib, "ws2_32.lib")  // Link with Winsock library

#define PORT 8080
#define BUFFER_SIZE 1024

int main() {
    WSADATA wsa;
    SOCKET server_fd, new_socket;
    struct sockaddr_in address;
    int addrlen = sizeof(address);
    char buffer[BUFFER_SIZE] = {0};

    // Initialize Winsock
    if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0) {
        printf("WSAStartup failed. Error Code: %d\n", WSAGetLastError());
        return 1;
    }
```

```
// Create socket
if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == INVALID_SOCKET) {
    printf("Socket creation failed. Error Code: %d\n", WSAGetLastError());
    return 1;
}

// Configure server address
address.sin_family = AF_INET;
address.sin_addr.s_addr = INADDR_ANY;
address.sin_port = htons(PORT);

// Bind socket
if (bind(server_fd, (struct sockaddr*)&address, sizeof(address)) == SOCKET_ERROR) {
    printf("Bind failed. Error Code: %d\n", WSAGetLastError());
    return 1;
}

// Listen for client
if (listen(server_fd, 3) == SOCKET_ERROR) {
    printf("Listen failed. Error Code: %d\n", WSAGetLastError());
    return 1;
}

printf("Server listening on port %d...\n", PORT);

// Accept client connection
if ((new_socket = accept(server_fd, (struct sockaddr*)&address, &addrlen)) == INVALID_SOCKET) {
    printf("Accept failed. Error Code: %d\n", WSAGetLastError());
    return 1;
}
```

```
    printf("Client connected.\n");

    while (1) {
        memset(buffer, 0, BUFFER_SIZE);
        int valread = recv(new_socket, buffer, BUFFER_SIZE, 0);
        if (valread <= 0) {
            printf("Client disconnected.\n");
            break;
        }

        printf("Client: %s", buffer);

        printf("Server: ");
        fgets(buffer, BUFFER_SIZE, stdin);
        send(new_socket, buffer, strlen(buffer), 0);
    }

    closesocket(new_socket);
    closesocket(server_fd);
    WSACleanup();
    return 0;
}
```