**Server code**

#include <iostream>

#include <cstring>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <unistd.h>

#define PORT 65432

#define BUFFER\_SIZE 1024

int main() {

int server\_fd, new\_socket;

struct sockaddr\_in address;

int addrlen = sizeof(address);

char buffer[BUFFER\_SIZE] = {0};

// Creating socket file descriptor

if ((server\_fd = socket(AF\_INET, SOCK\_STREAM, 0)) == 0) {

perror("socket failed");

exit(EXIT\_FAILURE);

}

// Define the server address

address.sin\_family = AF\_INET;

address.sin\_addr.s\_addr = inet\_addr("172.10.2.32");

address.sin\_port = htons(PORT);

// Bind the socket to the network address and port

if (bind(server\_fd, (struct sockaddr \*)&address, sizeof(address)) < 0) {

perror("bind failed");

close(server\_fd);

exit(EXIT\_FAILURE);

}

// Listen for incoming connections

if (listen(server\_fd, 3) < 0) {

perror("listen");

close(server\_fd);

exit(EXIT\_FAILURE);

}

std::cout << "Server listening on 172.10.2.32:" << PORT << std::endl;

// Accept a connection

if ((new\_socket = accept(server\_fd, (struct sockaddr )&address, (socklen\_t)&addrlen)) < 0) {

perror("accept");

close(server\_fd);

exit(EXIT\_FAILURE);

}

// Communicate with the client

while (true) {

memset(buffer, 0, BUFFER\_SIZE);

int valread = read(new\_socket, buffer, BUFFER\_SIZE);

if (valread <= 0) {

break;

}

std::cout << "Received from client: " << buffer << std::endl;

send(new\_socket, buffer, strlen(buffer), 0);

}

close(new\_socket);

close(server\_fd);

    return 0;

}

Client code:

#include <iostream>

#include <cstring>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <unistd.h>

#define PORT 65432

#define BUFFER\_SIZE 1024

int main() {

int sock = 0;

struct sockaddr\_in serv\_addr;

char buffer[BUFFER\_SIZE] = {0};

// Creating socket file descriptor

if ((sock = socket(AF\_INET, SOCK\_STREAM, 0)) < 0) {

std::cerr << "Socket creation error" << std::endl;

return -1;

}

// Define the server address

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_port = htons(PORT);

// Convert IPv4 address from text to binary form

if (inet\_pton(AF\_INET, "172.10.2.32", &serv\_addr.sin\_addr) <= 0) {

std::cerr << "Invalid address/ Address not supported" << std::endl;

return -1;

}

// Connect to the server

if (connect(sock, (struct sockaddr \*)&serv\_addr, sizeof(serv\_addr)) < 0) {

std::cerr << "Connection Failed" << std::endl;

return -1;

}

while (true) {

std::cout << "Enter message to send (type 'exit' to close): ";

std::string message;

std::getline(std::cin, message);

if (message == "exit") {

break;

}

send(sock, message.c\_str(), message.length(), 0);

std::cout << "Message sent" << std::endl;

memset(buffer, 0, BUFFER\_SIZE);

int valread = read(sock, buffer, BUFFER\_SIZE);

std::cout << "Received from server: " << buffer << std::endl;

}

close(sock);

    return 0;

}