# TCP ASSIGNMENT

1) client send a number to server and server check it is armstrong or not and send it to client or not

Server:  
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#include <math.h>

#define MYPORT 5679

#define BUFLEN 512

// Function to check Armstrong number

int isArmstrong(int num) {

int original = num, sum = 0, digits = 0, temp;

temp = num;

while (temp > 0) {

digits++;

temp /= 10;

}

temp = num;

while (temp > 0) {

int digit = temp % 10;

sum += pow(digit, digits);

temp /= 10;

}

return (sum == original);

}

int main() {

int sockfd, new\_fd;

struct sockaddr\_in my\_addr, client\_addr;

socklen\_t addr\_len = sizeof(client\_addr);

char buf[BUFLEN];

// Create TCP socket

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

if (sockfd == -1) {

perror("socket");

exit(EXIT\_FAILURE);

}

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

my\_addr.sin\_port = htons(MYPORT);

my\_addr.sin\_addr.s\_addr = INADDR\_ANY;

memset(&(my\_addr.sin\_zero), '\0', 8);

// Bind socket

if (bind(sockfd, (struct sockaddr \*)&my\_addr, sizeof(my\_addr)) == -1) {

perror("bind");

close(sockfd);

exit(EXIT\_FAILURE);

}

// Listen for connections

if (listen(sockfd, 1) == -1) {

perror("listen");

close(sockfd);

exit(EXIT\_FAILURE);

}

printf("Server is waiting for TCP connection...\n");

// Accept connection

new\_fd = accept(sockfd, (struct sockaddr \*)&client\_addr, &addr\_len);

if (new\_fd == -1) {

perror("accept");

close(sockfd);

exit(EXIT\_FAILURE);

}

// Receive number from client

ssize\_t recv\_len = recv(new\_fd, buf, BUFLEN - 1, 0);

if (recv\_len == -1) {

perror("recv");

close(new\_fd);

close(sockfd);

exit(EXIT\_FAILURE);

}

buf[recv\_len] = '\0';

int number = atoi(buf);

printf("Received number: %d\n", number);

// Check Armstrong

char result[BUFLEN];

if (isArmstrong(number)) {

snprintf(result, sizeof(result), "%d is an Armstrong number", number);

} else {

snprintf(result, sizeof(result), "%d is NOT an Armstrong number", number);

}

// Send result to client

if (send(new\_fd, result, strlen(result), 0) == -1) {

perror("send");

} else {

printf("Result sent to client: %s\n", result);

}

close(new\_fd);

close(sockfd);

return 0;

}

Client:  
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#define SERVER\_IP "127.0.0.1" // Change if server is remote

#define SERVER\_PORT 5679

#define BUFLEN 512

int main() {

int sockfd;

struct sockaddr\_in server\_addr;

char buf[BUFLEN];

// Create TCP socket

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

if (sockfd == -1) {

perror("socket");

exit(EXIT\_FAILURE);

}

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

server\_addr.sin\_port = htons(SERVER\_PORT);

server\_addr.sin\_addr.s\_addr = inet\_addr(SERVER\_IP);

memset(&(server\_addr.sin\_zero), '\0', 8);

// Connect to server

if (connect(sockfd, (struct sockaddr \*)&server\_addr, sizeof(server\_addr)) == -1) {

perror("connect");

close(sockfd);

exit(EXIT\_FAILURE);

}

// Input number from user

int number;

printf("Enter a number: ");

scanf("%d", &number);

char message[BUFLEN];

snprintf(message, sizeof(message), "%d", number);

// Send number to server

if (send(sockfd, message, strlen(message), 0) == -1) {

perror("send");

close(sockfd);

exit(EXIT\_FAILURE);

}

printf("Number sent to server: %s\n", message);

// Receive response from server

ssize\_t recv\_len = recv(sockfd, buf, BUFLEN - 1, 0);

if (recv\_len == -1) {

perror("recv");

close(sockfd);

exit(EXIT\_FAILURE);

}

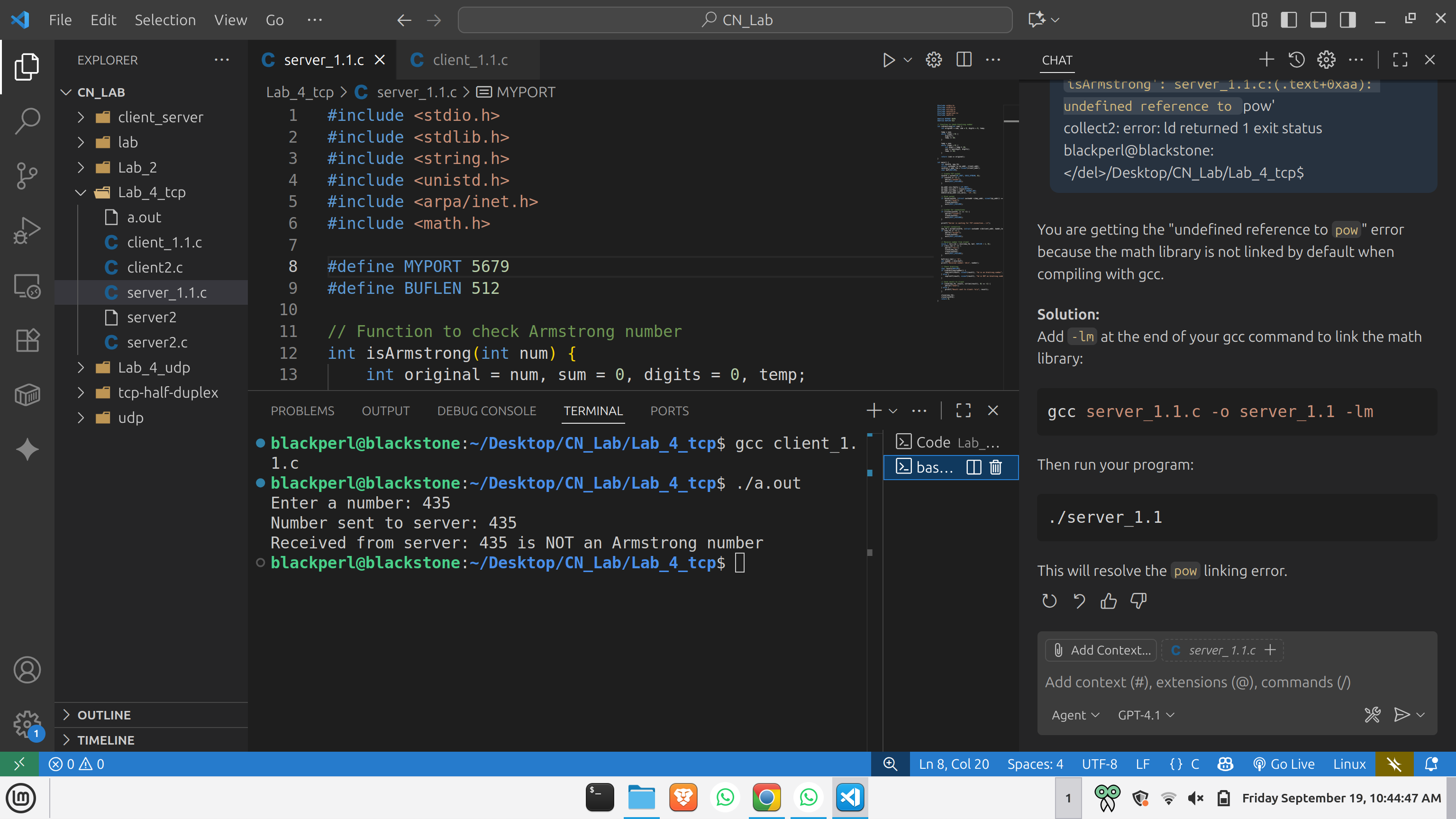
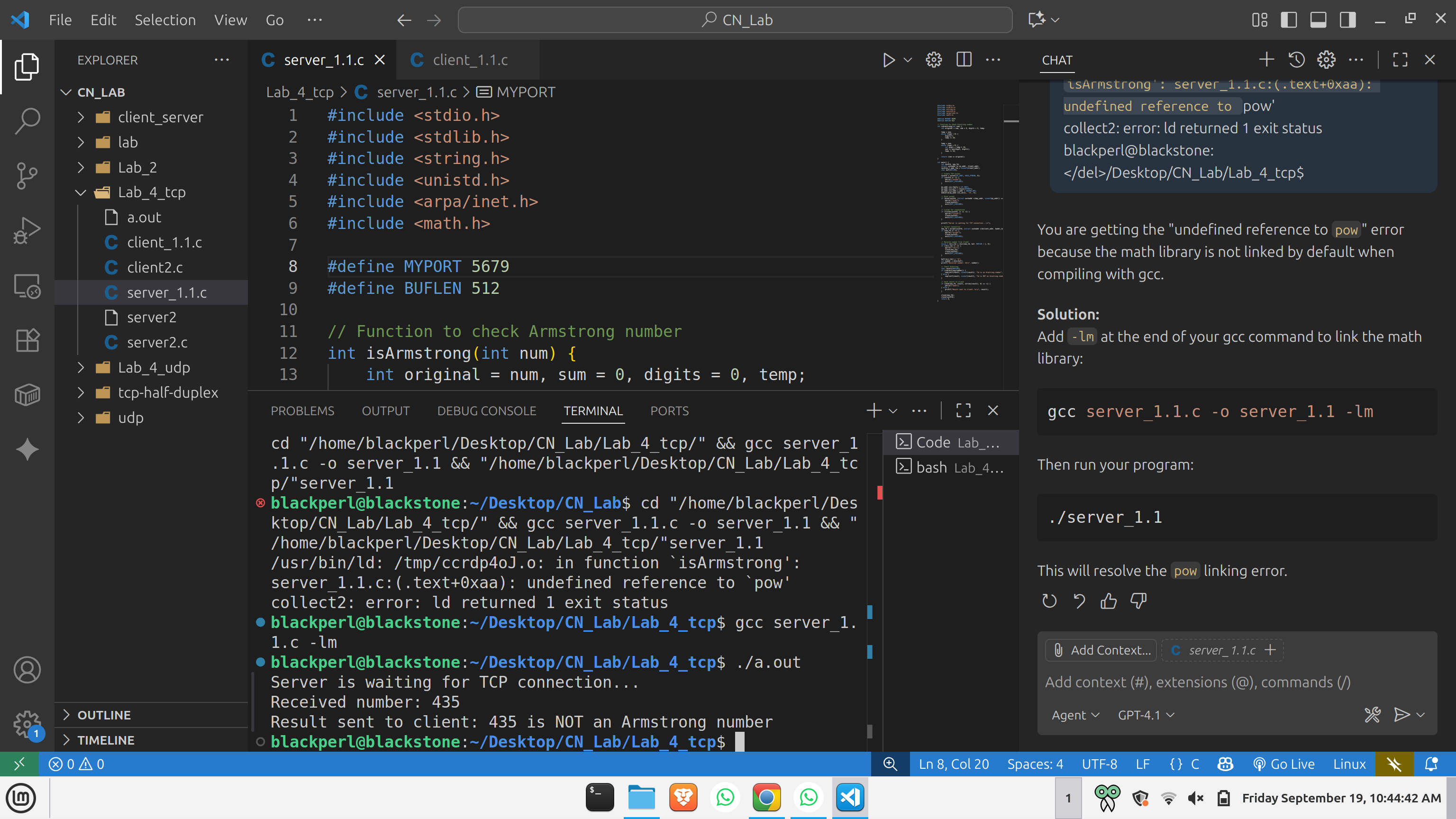
buf[recv\_len] = '\0';

printf("Received from server: %s\n", buf);

close(sockfd);

return 0;

}

Output:  


2) two different client send 2 numbers to the server, server will exchange the 2 numbers between clients

Server:  
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#define MYPORT 5678

#define BUFLEN 512

int main() {

int sockfd, client1\_fd, client2\_fd;

struct sockaddr\_in my\_addr, client\_addr;

socklen\_t addr\_len = sizeof(client\_addr);

char buf1[BUFLEN], buf2[BUFLEN];

// Create TCP socket

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

if (sockfd == -1) {

perror("socket");

exit(EXIT\_FAILURE);

}

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

my\_addr.sin\_port = htons(MYPORT);

my\_addr.sin\_addr.s\_addr = INADDR\_ANY;

memset(&(my\_addr.sin\_zero), '\0', 8);

// Bind socket

if (bind(sockfd, (struct sockaddr \*)&my\_addr, sizeof(my\_addr)) == -1) {

perror("bind");

close(sockfd);

exit(EXIT\_FAILURE);

}

// Listen for connections

if (listen(sockfd, 2) == -1) {

perror("listen");

close(sockfd);

exit(EXIT\_FAILURE);

}

printf("Server waiting for 2 clients...\n");

// Accept first client

client1\_fd = accept(sockfd, (struct sockaddr \*)&client\_addr, &addr\_len);

if (client1\_fd == -1) {

perror("accept client1");

close(sockfd);

exit(EXIT\_FAILURE);

}

printf("Client 1 connected!\n");

// Accept second client

client2\_fd = accept(sockfd, (struct sockaddr \*)&client\_addr, &addr\_len);

if (client2\_fd == -1) {

perror("accept client2");

close(client1\_fd);

close(sockfd);

exit(EXIT\_FAILURE);

}

printf("Client 2 connected!\n");

// Receive number from client 1

ssize\_t recv\_len1 = recv(client1\_fd, buf1, BUFLEN - 1, 0);

if (recv\_len1 <= 0) {

perror("recv client1");

close(client1\_fd);

close(client2\_fd);

close(sockfd);

exit(EXIT\_FAILURE);

}

buf1[recv\_len1] = '\0';

printf("Received from Client 1: %s\n", buf1);

// Receive number from client 2

ssize\_t recv\_len2 = recv(client2\_fd, buf2, BUFLEN - 1, 0);

if (recv\_len2 <= 0) {

perror("recv client2");

close(client1\_fd);

close(client2\_fd);

close(sockfd);

exit(EXIT\_FAILURE);

}

buf2[recv\_len2] = '\0';

printf("Received from Client 2: %s\n", buf2);

// Exchange numbers

if (send(client1\_fd, buf2, strlen(buf2), 0) == -1) {

perror("send to client1");

} else {

printf("Sent to Client 1: %s\n", buf2);

}

if (send(client2\_fd, buf1, strlen(buf1), 0) == -1) {

perror("send to client2");

} else {

printf("Sent to Client 2: %s\n", buf1);

}

close(client1\_fd);

close(client2\_fd);

close(sockfd);

return 0;

}

Client:  
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#define SERVER\_IP "127.0.0.1" // Change if server is remote

#define SERVER\_PORT 5678

#define BUFLEN 512

int main() {

int sockfd;

struct sockaddr\_in server\_addr;

char buf[BUFLEN];

// Create TCP socket

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

if (sockfd == -1) {

perror("socket");

exit(EXIT\_FAILURE);

}

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

server\_addr.sin\_port = htons(SERVER\_PORT);

server\_addr.sin\_addr.s\_addr = inet\_addr(SERVER\_IP);

memset(&(server\_addr.sin\_zero), '\0', 8);

// Connect to server

if (connect(sockfd, (struct sockaddr \*)&server\_addr, sizeof(server\_addr)) == -1) {

perror("connect");

close(sockfd);

exit(EXIT\_FAILURE);

}

// Input number from user

int number;

printf("Enter a number: ");

scanf("%d", &number);

char message[BUFLEN];

snprintf(message, sizeof(message), "%d", number);

// Send number to server

if (send(sockfd, message, strlen(message), 0) == -1) {

perror("send");

close(sockfd);

exit(EXIT\_FAILURE);

}

printf("Number sent: %s\n", message);

// Receive exchanged number

ssize\_t recv\_len = recv(sockfd, buf, BUFLEN - 1, 0);

if (recv\_len <= 0) {

perror("recv");

close(sockfd);

exit(EXIT\_FAILURE);

}

buf[recv\_len] = '\0';

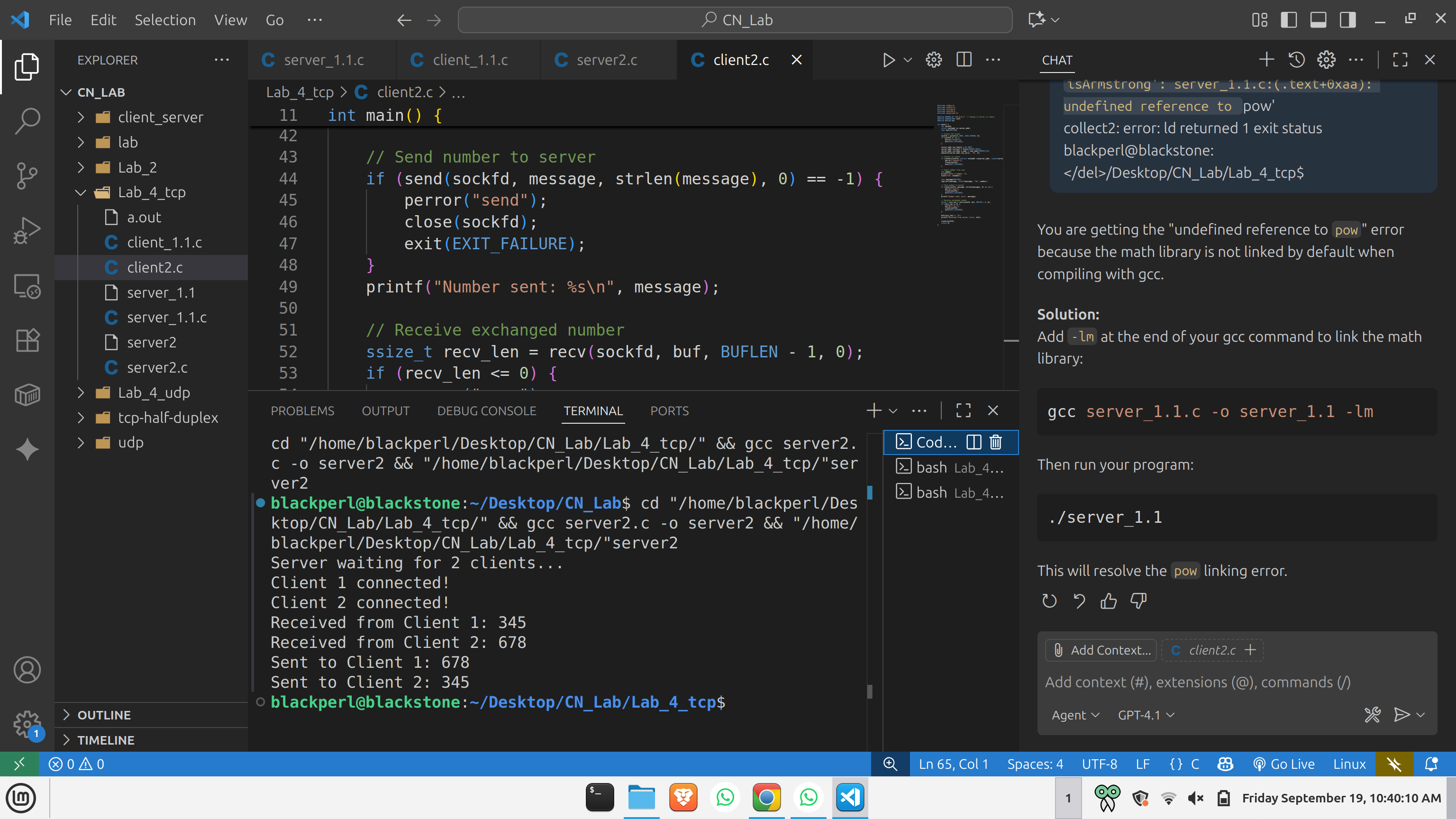
printf("Received from server: %s\n", buf);

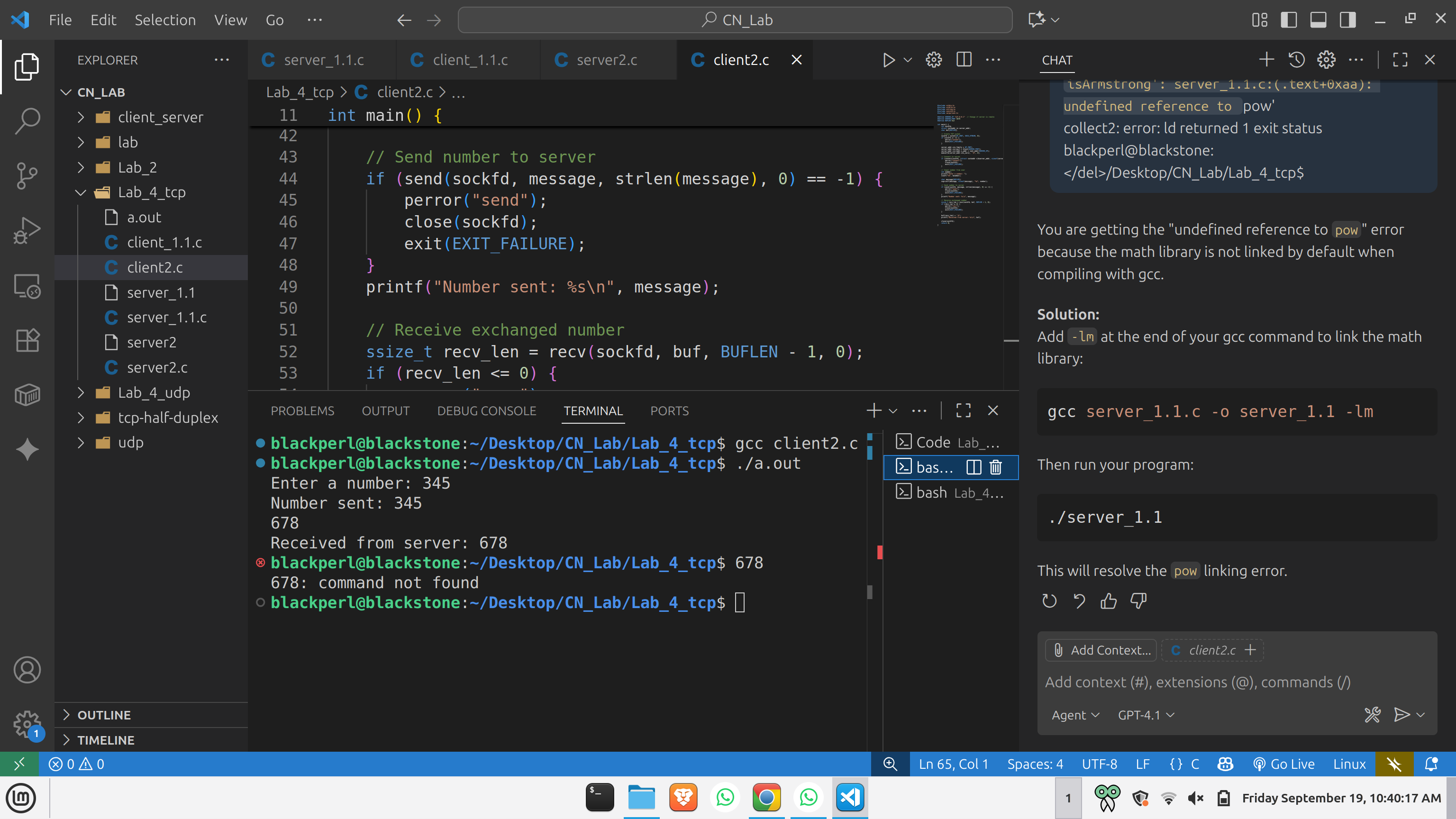
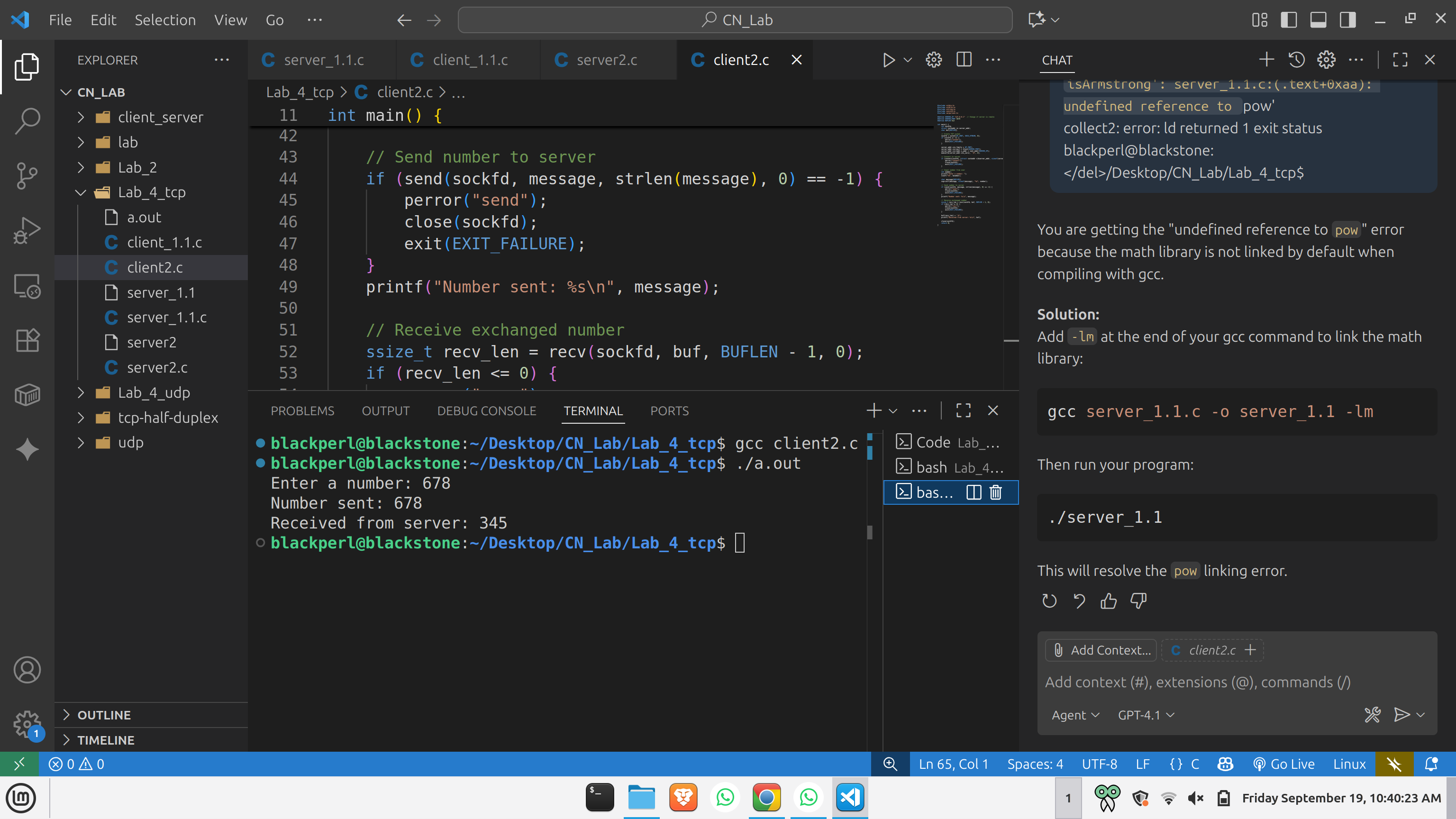
close(sockfd);

return 0;

}

Output:



  
  
  
  
  
  
  
  
  
  
  
3) Server has a student database which has roll no, name and marks in Structure form and store atleast 5 records , client send a roll no to server and server return the details information of this particular rollno to the client.  
  
Server:  
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#define MYPORT 5678

#define BUFLEN 512

// Define student structure

struct Student {

int rollno;

char name[50];

int marks;

};

// Demo student database

struct Student database[] = {

{1, "Alice", 85},

{2, "Bob", 90},

{3, "Charlie", 78},

{4, "David", 88},

{5, "Eve", 92}

};

int db\_size = 5;

int main() {

int sockfd, new\_fd;

struct sockaddr\_in my\_addr, client\_addr;

socklen\_t addr\_len = sizeof(client\_addr);

char buf[BUFLEN];

// Create TCP socket

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

if (sockfd == -1) {

perror("socket");

exit(EXIT\_FAILURE);

}

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

my\_addr.sin\_port = htons(MYPORT);

my\_addr.sin\_addr.s\_addr = INADDR\_ANY;

memset(&(my\_addr.sin\_zero), '\0', 8);

// Bind socket

if (bind(sockfd, (struct sockaddr \*)&my\_addr, sizeof(my\_addr)) == -1) {

perror("bind");

close(sockfd);

exit(EXIT\_FAILURE);

}

// Listen for connections

if (listen(sockfd, 1) == -1) {

perror("listen");

close(sockfd);

exit(EXIT\_FAILURE);

}

printf("Server is waiting for TCP connection...\n");

// Accept connection

new\_fd = accept(sockfd, (struct sockaddr \*)&client\_addr, &addr\_len);

if (new\_fd == -1) {

perror("accept");

close(sockfd);

exit(EXIT\_FAILURE);

}

// Receive roll number from client

ssize\_t recv\_len = recv(new\_fd, buf, BUFLEN - 1, 0);

if (recv\_len <= 0) {

perror("recv");

close(new\_fd);

close(sockfd);

exit(EXIT\_FAILURE);

}

buf[recv\_len] = '\0';

int roll = atoi(buf);

printf("Received roll no: %d\n", roll);

// Search student in database

char result[BUFLEN];

int found = 0;

for (int i = 0; i < db\_size; i++) {

if (database[i].rollno == roll) {

snprintf(result, sizeof(result), "Roll: %d, Name: %s, Marks: %d",

database[i].rollno, database[i].name, database[i].marks);

found = 1;

break;

}

}

if (!found) {

snprintf(result, sizeof(result), "Roll no %d not found!", roll);

}

// Send response to client

if (send(new\_fd, result, strlen(result), 0) == -1) {

perror("send");

} else {

printf("Sent to client: %s\n", result);

}

close(new\_fd);

close(sockfd);

return 0;

}

Client:  
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#define SERVER\_IP "127.0.0.1" // Change if server is remote

#define SERVER\_PORT 5678

#define BUFLEN 512

int main() {

int sockfd;

struct sockaddr\_in server\_addr;

char buf[BUFLEN];

// Create TCP socket

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

if (sockfd == -1) {

perror("socket");

exit(EXIT\_FAILURE);

}

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

server\_addr.sin\_port = htons(SERVER\_PORT);

server\_addr.sin\_addr.s\_addr = inet\_addr(SERVER\_IP);

memset(&(server\_addr.sin\_zero), '\0', 8);

// Connect to server

if (connect(sockfd, (struct sockaddr \*)&server\_addr, sizeof(server\_addr)) == -1) {

perror("connect");

close(sockfd);

exit(EXIT\_FAILURE);

}

// Input roll number

int roll;

printf("Enter roll number to search: ");

scanf("%d", &roll);

char message[BUFLEN];

snprintf(message, sizeof(message), "%d", roll);

// Send roll number to server

if (send(sockfd, message, strlen(message), 0) == -1) {

perror("send");

close(sockfd);

exit(EXIT\_FAILURE);

}

printf("Roll number sent: %s\n", message);

// Receive student details from server

ssize\_t recv\_len = recv(sockfd, buf, BUFLEN - 1, 0);

if (recv\_len <= 0) {

perror("recv");

close(sockfd);

exit(EXIT\_FAILURE);

}

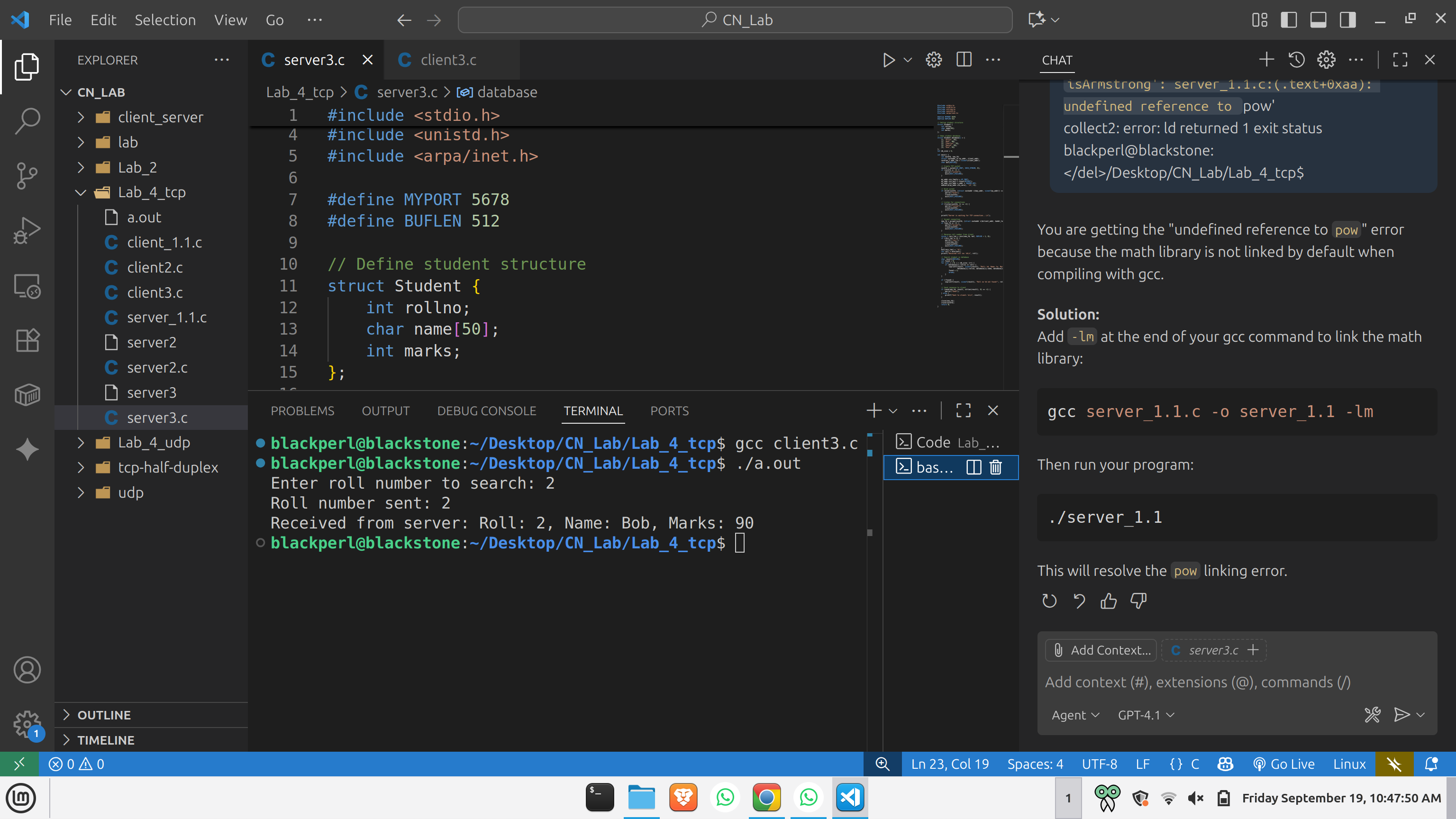
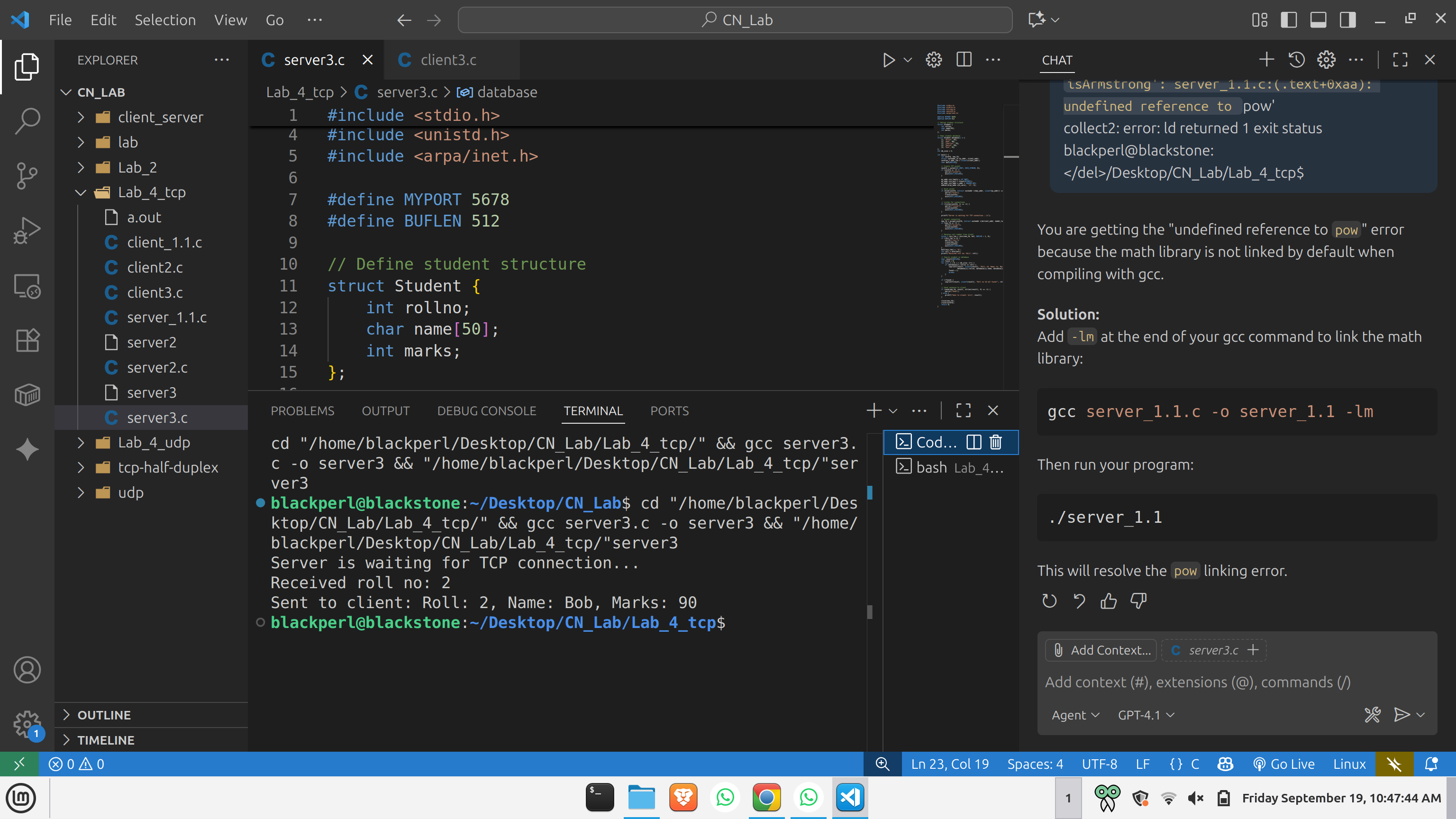
buf[recv\_len] = '\0';

printf("Received from server: %s\n", buf);

close(sockfd);

return 0;

}

Output: