# UDP ASSIGNMENT

Q1.) Write a client Server UDP program client will send the 3 numbers to server and server will send the sum of that input given.

Ans:

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;

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

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

char buf[BUFLEN];

int num1, num2, num3, sum;

// Create UDP socket

sockfd = socket(AF\_INET, SOCK\_DGRAM, 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);

}

printf("UDP Server is waiting for data...\n");

// Receive message from client

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

(struct sockaddr \*)&client\_addr, &addr\_len);

if (recv\_len == -1) {

perror("recvfrom");

close(sockfd);

exit(EXIT\_FAILURE);

}

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

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

// Parse numbers and compute sum

sscanf(buf, "%d %d %d", &num1, &num2, &num3);

sum = num1 + num2 + num3;

// Send result back to client

char result[BUFLEN];

snprintf(result, sizeof(result), "Sum = %d", sum);

if (sendto(sockfd, result, strlen(result), 0,

(struct sockaddr \*)&client\_addr, addr\_len) == -1) {

perror("sendto");

close(sockfd);

exit(EXIT\_FAILURE);

}

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

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];

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

// Create UDP socket

sockfd = socket(AF\_INET, SOCK\_DGRAM, 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);

// Input numbers from user

int a, b, c;

printf("Enter 3 numbers: ");

scanf("%d %d %d", &a, &b, &c);

char message[BUFLEN];

snprintf(message, sizeof(message), "%d %d %d", a, b, c);

// Send message to server

if (sendto(sockfd, message, strlen(message), 0,

(struct sockaddr \*)&server\_addr, addr\_len) == -1) {

perror("sendto");

close(sockfd);

exit(EXIT\_FAILURE);

}

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

// Receive response from server

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

(struct sockaddr \*)&server\_addr, &addr\_len);

if (recv\_len == -1) {

perror("recvfrom");

close(sockfd);

exit(EXIT\_FAILURE);

}

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

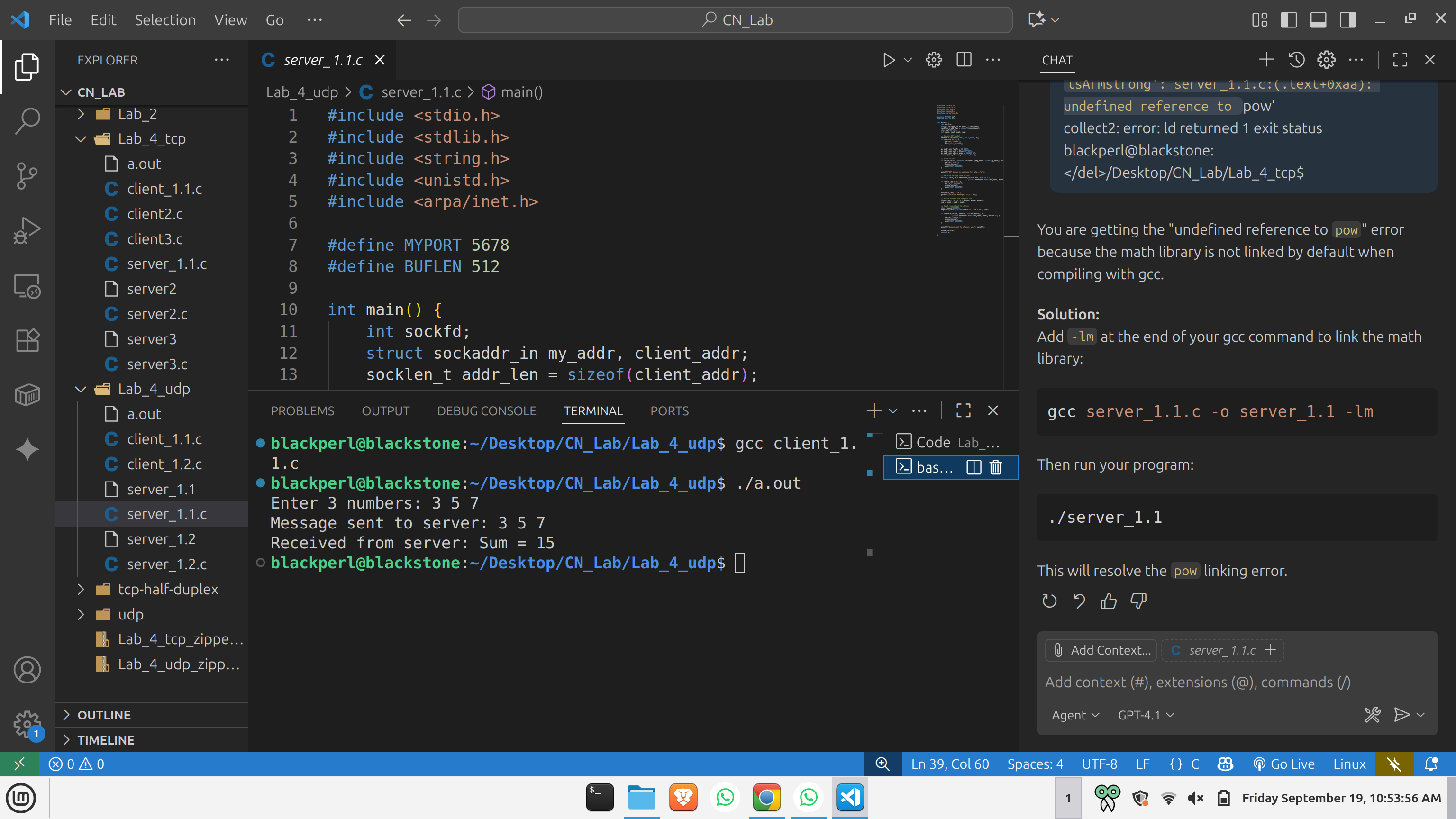
printf("Received from server: %s\n", buf);

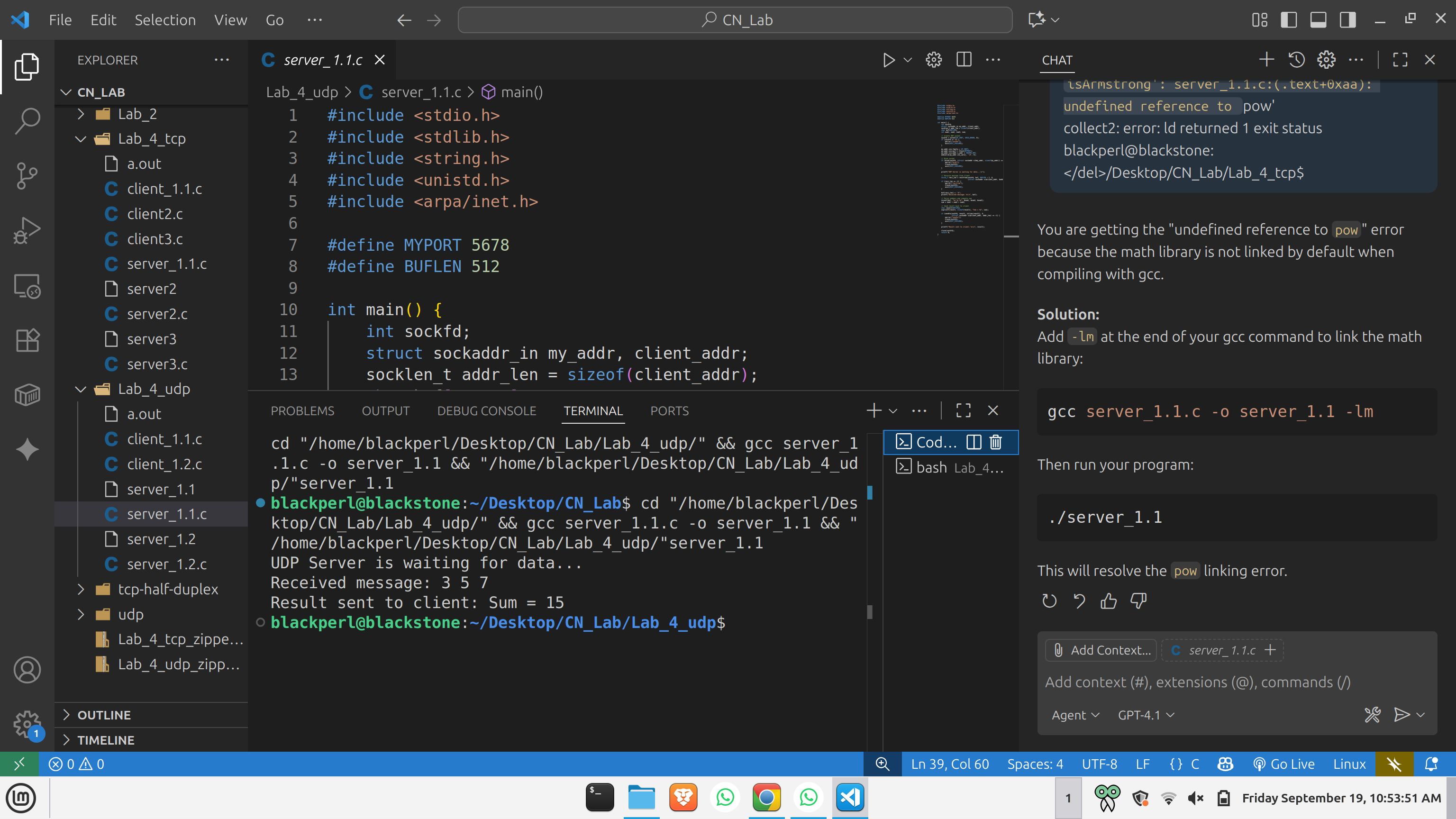
close(sockfd);

return 0;

}

Outputs:





Q2.) Write a client Server UDP program client will send the string to the server and server will send it's in reverse order 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

// Function to reverse a string in place

void reverse\_string(char \*str) {

int i, j;

char temp;

int len = strlen(str);

for (i = 0, j = len - 1; i < j; i++, j--) {

temp = str[i];

str[i] = str[j];

str[j] = temp;

}

}

int main() {

int sockfd;

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

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

char buf[BUFLEN];

// Create UDP socket

sockfd = socket(AF\_INET, SOCK\_DGRAM, 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);

}

printf("UDP Server is waiting for data...\n");

// Receive string from client

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

(struct sockaddr \*)&client\_addr, &addr\_len);

if (recv\_len == -1) {

perror("recvfrom");

close(sockfd);

exit(EXIT\_FAILURE);

}

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

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

// Reverse the string

reverse\_string(buf);

// Send reversed string back to client

if (sendto(sockfd, buf, strlen(buf), 0,

(struct sockaddr \*)&client\_addr, addr\_len) == -1) {

perror("sendto");

close(sockfd);

exit(EXIT\_FAILURE);

}

printf("Reversed string sent back: %s\n", buf);

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];

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

// Create UDP socket

sockfd = socket(AF\_INET, SOCK\_DGRAM, 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);

// Input string from user

char message[BUFLEN];

printf("Enter a string: ");

fgets(message, sizeof(message), stdin);

message[strcspn(message, "\n")] = '\0'; // Remove newline

// Send string to server

if (sendto(sockfd, message, strlen(message), 0,

(struct sockaddr \*)&server\_addr, addr\_len) == -1) {

perror("sendto");

close(sockfd);

exit(EXIT\_FAILURE);

}

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

// Receive reversed string from server

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

(struct sockaddr \*)&server\_addr, &addr\_len);

if (recv\_len == -1) {

perror("recvfrom");

close(sockfd);

exit(EXIT\_FAILURE);

}

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

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

close(sockfd);

return 0;

}

Outputs:

