Name : Rajvardhan Reddy
Reg No : 180905093
Sec : B
Roll No : 19

CN Lab Session1: <u>Socket programming</u>

P1) Write a c program to demonstrate the working of UDP echo Client/Server.

Server program :

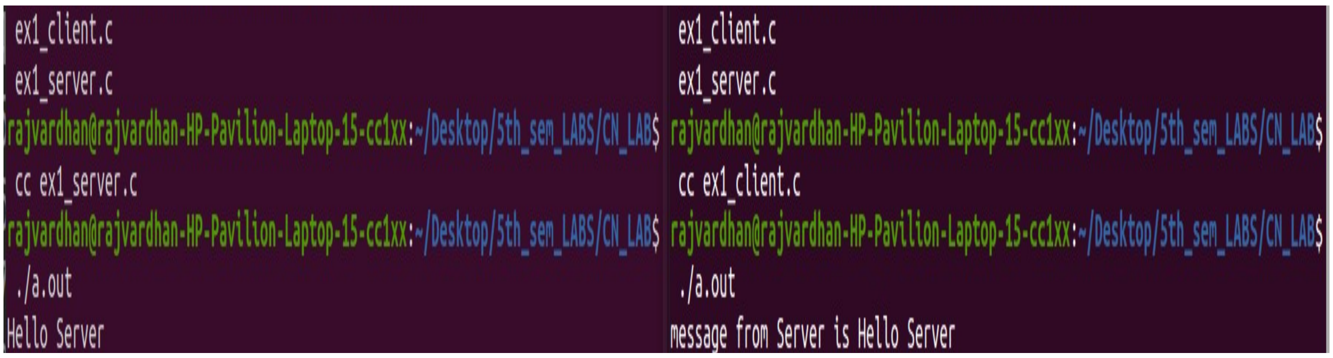
```
// server program for udp connection
#include <stdio.h>
#include <strings.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include<netinet/in.h>
#define PORT 5000
#define MAXLINE 1000

int main()
{
	char buffer[100];
	int servsockfd, len,n;
	struct sockaddr_in servaddr, cliaddr;
	bzero(&servaddr, sizeof(servaddr));

	// Create a UDP Socket
	servsockfd = socket(AF_INET, SOCK_DGRAM, 0);
	servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
	servaddr.sin_port = htons(PORT);
	servaddr.sin_family = AF_INET;

	// bind server address to socket descriptor
	bind(servsockfd, (struct sockaddr*)&servaddr, sizeof(servaddr));

	//receive the datagram
	len = sizeof(cliaddr);
	n = recvfrom(servsockfd, buffer, sizeof(buffer),0, (struct
sockaddr*)&cliaddr,&len);
	buffer[n] = '\0';
```

```
          puts(buffer);
//Echoing back to the client
            sendto(servsockfd, buffer, n, 0, (struct sockaddr*)&cliaddr, sizeof(cliaddr));
                getchar();
}
```

Client program :

```
// udp client driver program

#include <stdio.h>
#include <strings.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include<netinet/in.h>
#include<unistd.h>
#include<stdlib.h>

#define PORT 5000
#define MAXLINE 1000

// Driver code
int main()
{
          char buffer[100];
          char *message = "Hello Server";
          int sockfd, n,len;
          struct sockaddr_in servaddr, cliaddr;

          // clear servaddr
          bzero(&servaddr, sizeof(servaddr));
          servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
          servaddr.sin_port = htons(PORT);
          servaddr.sin_family = AF_INET;

          // create datagram socket
          sockfd = socket(AF_INET, SOCK_DGRAM, 0);
          sendto(sockfd, message, MAXLINE, 0, (struct sockaddr*)&servaddr,
sizeof(servaddr));
          len=sizeof(cliaddr);
          // waiting for response
          n=recvfrom(sockfd, buffer, sizeof(buffer), 0, (struct
sockaddr*)&cliaddr,&len );
        buffer[n]='\0';
          printf("message from Server is %s \n",buffer);
        getchar();

          // close the descriptor
          close(sockfd);
}
```

Output :



```
ex1_client.c
ex1_server.c
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CN_LAB$
cc ex1_server.c
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CN_LAB$
./a.out
Hello Server
```

```
ex1_client.c
ex1_server.c
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CN_LAB$
cc ex1_client.c
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CN_LAB$
./a.out
message from Server is Hello Server
```

P2) Write a c program to demonstrate the working of TCP client server as follows: After connection set up client send a message. Server will reply to this. If server decides to close the program then it will send a message exit to client then closes itself. Client will close after receiving this message.. (Note: In each program there is a function that handles the client and server function and main program is responsible for socket creation and connection setup.)

Server program :

```
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#define MAX 80
#define PORT 8080
#define SA struct sockaddr

// Function designed for chat between client and server.
```

```
void servfunc(int sockfd)
{
        char buff[MAX];
        int n;
        // infinite loop for chat
        for (;;) {
                bzero(buff, MAX);

                // read the message from client and copy it in buffer
                read(sockfd, buff, sizeof(buff));
                // print buffer which contains the client contents
                printf("From client: %s\t To client : ", buff);

                bzero(buff, sizeof(buff));
// Read server message from keyboard in the buffer
                n=0;
                while ((buff[n++] = getchar()) != '\n')
                        ;
// and send that buffer to client
                write(sockfd, buff, sizeof(buff));

                // if msg contains "Exit" then server exit and session ended.
                if (strncmp("exit", buff, 4) == 0) {
                        printf("Server Exit...\n");
                        break;
                }
        }
}

// Driver function
int main()
{
        int sockfd, connfd, len;
        struct sockaddr_in servaddr, cli;
```

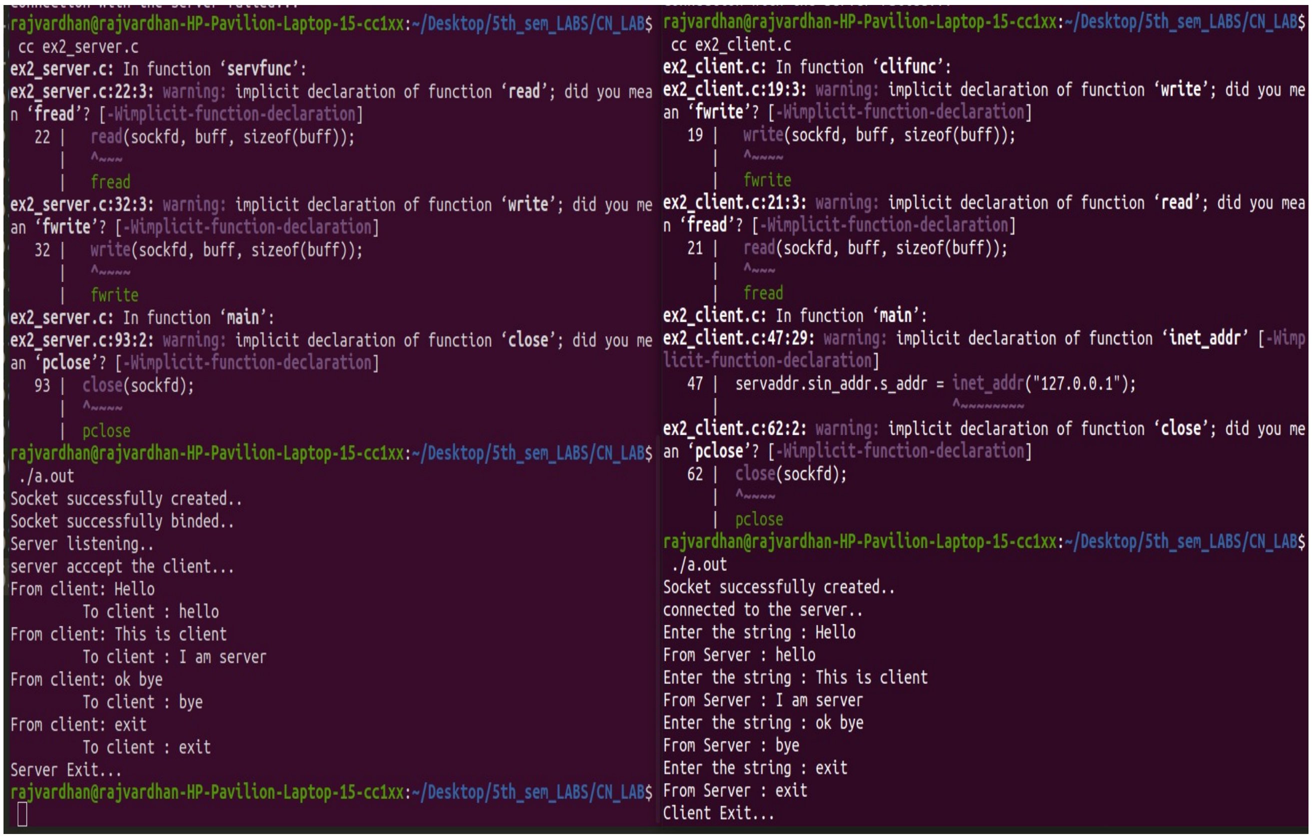
```
// socket create and verification
sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd == -1) {
        printf("socket creation failed...\n");
        exit(0);
}
else
        printf("Socket successfully created..\n");
bzero(&servaddr, sizeof(servaddr));

// assign IP, PORT
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
servaddr.sin_port = htons(PORT);

// Binding newly created socket to given IP and verification
if ((bind(sockfd, (SA*)&servaddr, sizeof(servaddr))) != 0) {
        printf("socket bind failed...\n");
        exit(0);
}
else
        printf("Socket successfully binded..\n");

// Now server is ready to listen and verification
if ((listen(sockfd, 5)) != 0) {
        printf("Listen failed...\n");
        exit(0);
}
else
        printf("Server listening..\n");
len = sizeof(cli);

// Accept the data packet from client and verification
```

```
    connfd = accept(sockfd, (SA*)&cli, &len);
    if (connfd < 0) {
        printf("server acccept failed...\n");
        exit(0);
    }
    else
        printf("server acccept the client...\n");

    // Function for chatting between client and server
    servfunc(connfd);

    // After sending exit message close the socket
    close(sockfd);
```

Client program :

```
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#define MAX 80
#define PORT 8080
#define SA struct sockaddr
void clifunc(int sockfd)
{
    char buff[MAX];
    int n;
    for (;;) {
        bzero(buff, sizeof(buff));
        printf("Enter the string : ");
        n = 0;
        while ((buff[n++] = getchar()) != '\n')
            ;
        write(sockfd, buff, sizeof(buff));
        bzero(buff, sizeof(buff));
        read(sockfd, buff, sizeof(buff));
        printf("From Server : %s", buff);
        if ((strncmp(buff, "exit", 4)) == 0) {
            printf("Client Exit...\n");
            break;
        }
    }
}

int main()
{
    int sockfd, connfd;
    struct sockaddr_in servaddr, cli;

    // socket create and verification
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd == -1) {
        printf("socket creation failed...\n");
        exit(0);
    }
    else
        printf("Socket successfully created..\n");
    bzero(&servaddr, sizeof(servaddr));

    // assign IP, PORT
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    servaddr.sin_port = htons(PORT);

    // connect the client socket to server socket
    if (connect(sockfd, (SA*)&servaddr, sizeof(servaddr)) != 0) {
        printf("connection with the server failed...\n");
        exit(0);
    }
    else
        printf("connected to the server..\n");

    // function for client
    clifunc(sockfd);

    // close the socket
    close(sockfd);
}
```

Output :



```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CN_LAB$
 cc ex2_server.c
ex2_server.c: In function ‘servfunc’:
ex2_server.c:22:3: warning: implicit declaration of function ‘read’; did you mea
n ‘fread’? [-Wimplicit-function-declaration]
   22 |   read(sockfd, buff, sizeof(buff));
      |   ^~~~
      |   fread
ex2_server.c:32:3: warning: implicit declaration of function ‘write’; did you me
an ‘fwrite’? [-Wimplicit-function-declaration]
   32 |   write(sockfd, buff, sizeof(buff));
      |   ^~~~~
      |   fwrite
ex2_server.c: In function ‘main’:
ex2_server.c:93:2: warning: implicit declaration of function ‘close’; did you me
an ‘pclose’? [-Wimplicit-function-declaration]
   93 |  close(sockfd);
      |  ^~~~~
      |  pclose
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CN_LAB$
 ./a.out
Socket successfully created..
Socket successfully binded..
Server listening..
server acccept the client...
From client: Hello
         To client : hello
From client: This is client
         To client : I am server
From client: ok bye
         To client : bye
From client: exit
         To client : exit
Server Exit...
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CN_LAB$
```

```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CN_LAB$
 cc ex2_client.c
ex2_client.c: In function ‘clifunc’:
ex2_client.c:19:3: warning: implicit declaration of function ‘write’; did you me
an ‘fwrite’? [-Wimplicit-function-declaration]
   19 |   write(sockfd, buff, sizeof(buff));
      |   ^~~~~
      |   fwrite
ex2_client.c:21:3: warning: implicit declaration of function ‘read’; did you mea
n ‘fread’? [-Wimplicit-function-declaration]
   21 |   read(sockfd, buff, sizeof(buff));
      |   ^~~~
      |   fread
ex2_client.c: In function ‘main’:
ex2_client.c:47:29: warning: implicit declaration of function ‘inet_addr’ [-Wimp
licit-function-declaration]
   47 |  servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
      |                             ^~~~~~~~~
ex2_client.c:62:2: warning: implicit declaration of function ‘close’; did you me
an ‘pclose’? [-Wimplicit-function-declaration]
   62 |  close(sockfd);
      |  ^~~~~
      |  pclose
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CN_LAB$
 ./a.out
Socket successfully created..
connected to the server..
Enter the string : Hello
From Server : hello
Enter the string : This is client
From Server : I am server
Enter the string : ok bye
From Server : bye
Enter the string : exit
From Server : exit
Client Exit...
```

P3) Write a UDP client-server program where client sends rows of a matrix to the server combines them together as a two dimensional matrix and display the same.

Server program :

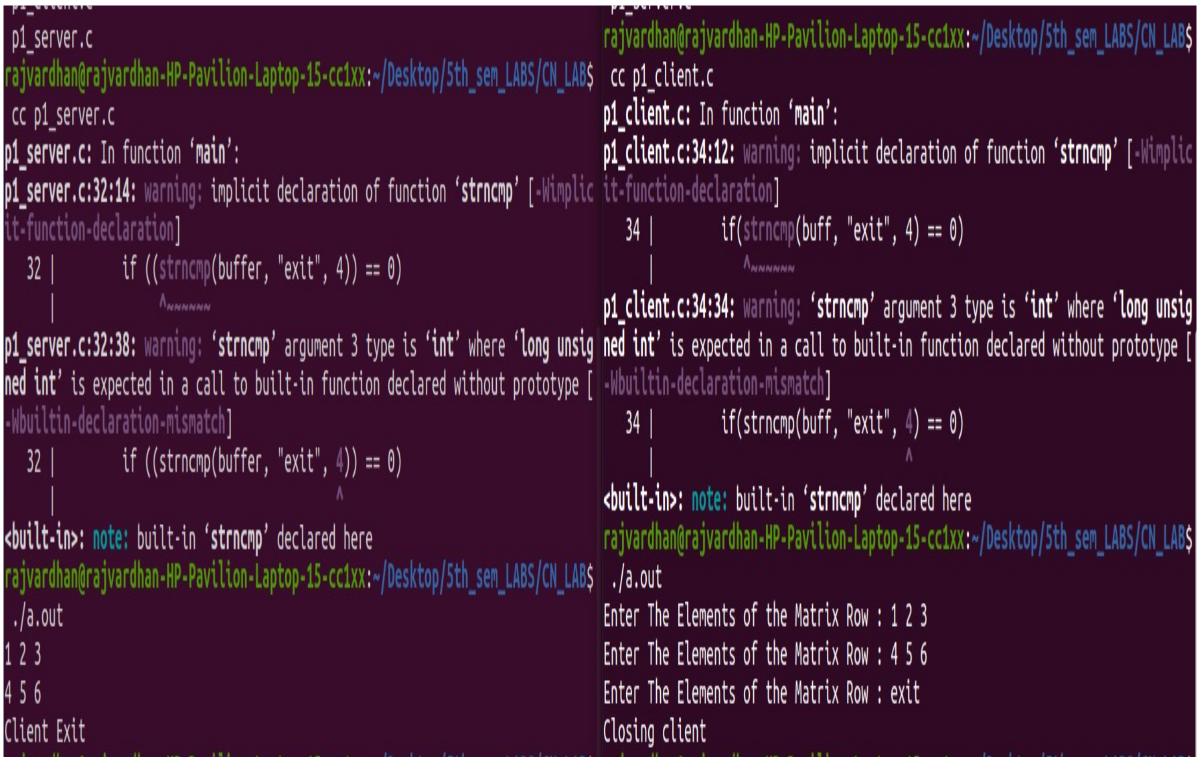
```
#include <stdio.h>

#include <strings.h>

#include <sys/types.h>

#include <arpa/inet.h>

#include <sys/socket.h>

#include <netinet/in.h>

#define PORT 5000

#define MAXLINE 1000

// Server code

int main()

{

    char buffer[100];

    int servsockfd, len, n;

    struct sockaddr_in servaddr, cliaddr;

    bzero(&servaddr, sizeof(servaddr));

    // Create a UDP Socket

    servsockfd = socket(AF_INET, SOCK_DGRAM, 0);

    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);

    servaddr.sin_port = htons(PORT);

    servaddr.sin_family = AF_INET;

    // bind server address to socket descriptor

    bind(servsockfd, (struct sockaddr *)&servaddr, sizeof(servaddr));

    //receive the datagram

    while(1)

    {

        bzero(buffer, sizeof(buffer));

        len = sizeof(cliaddr);

          n = recvfrom(servsockfd, buffer, sizeof(buffer), 0, (struct sockaddr *)&cliaddr,
&len);
```

```
        // buffer[n] = '\0';
        //Echoing back to the client
        if ((strncmp(buffer, "exit", 4)) == 0)
        {
           printf("Client Exit\n");
           break;
        }
        for(int i = 0; i < n; i++)
           printf("%c", buffer[i]);
        // sendto(servsockfd, buffer, n, 0, (struct sockaddr *)&cliaddr, sizeof(cliaddr));
        // getchar();
    }
    // close the descriptor
    // close(servsockfd);
}
```

Client program :

```
#include <stdio.h>
#include <strings.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>
#include <stdlib.h>
#define PORT 5000
#define MAXLINE 1000
// Driver code
int main()
{
    char buffer[100];
    //char *message = "";
    int sockfd, n, len;
    struct sockaddr_in servaddr, cliaddr;
    // clear servaddr
    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port = htons(PORT);
    servaddr.sin_family = AF_INET;
    // create datagram socket
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    char buff[100];
    for(;;)
    {
        bzero(buff, sizeof(buff));
        printf("Enter The Elements of the Matrix Row : ");
        n = 0;
        while ((buff[n++] = getchar()) != '\n');
        buff[n] = '\0';
            sendto(sockfd, &buff, MAXLINE, 0, (struct sockaddr *)&servaddr,
sizeof(servaddr));
        if(strncmp(buff, "exit", 4) == 0)
        {
            printf("Closing client\n");
            break;
        }
        bzero(buff, sizeof(buff));
        len = sizeof(cliaddr);
    }
    // close the descriptor
    close(sockfd);
}
```

Output :



```
 p1_server.c
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CN_LAB$
 cc p1_server.c
p1_server.c: In function ‘main’:
p1_server.c:32:14: warning: implicit declaration of function ‘strncmp’ [-Wimplic
it-function-declaration]
   32 |        if ((strncmp(buffer, "exit", 4)) == 0)
      |             ^~~~~~~
p1_server.c:32:38: warning: ‘strncmp’ argument 3 type is ‘int’ where ‘long unsig
ned int’ is expected in a call to built-in function declared without prototype [
-Wbuiltin-declaration-mismatch]
   32 |        if ((strncmp(buffer, "exit", 4)) == 0)
      |                                     ^
<built-in>: note: built-in ‘strncmp’ declared here
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CN_LAB$
 ./a.out
1 2 3
4 5 6
Client Exit
```

```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CN_LAB$
 cc p1_client.c
p1_client.c: In function ‘main’:
p1_client.c:34:12: warning: implicit declaration of function ‘strncmp’ [-Wimplic
it-function-declaration]
   34 |        if(strncmp(buff, "exit", 4) == 0)
      |           ^~~~~~~
p1_client.c:34:34: warning: ‘strncmp’ argument 3 type is ‘int’ where ‘long unsig
ned int’ is expected in a call to built-in function declared without prototype [
-Wbuiltin-declaration-mismatch]
   34 |        if(strncmp(buff, "exit", 4) == 0)
      |                                 ^
<built-in>: note: built-in ‘strncmp’ declared here
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CN_LAB$
 ./a.out
Enter The Elements of the Matrix Row : 1 2 3
Enter The Elements of the Matrix Row : 4 5 6
Enter The Elements of the Matrix Row : exit
Closing client
```

P4) Write a TCP client which sends a string to a server program. Server displays the string along with client IP and ephemeral port number. Server then responds to the client by echoing back the string in uppercase. The process continues until one of them types “QUIT”.

Server program :

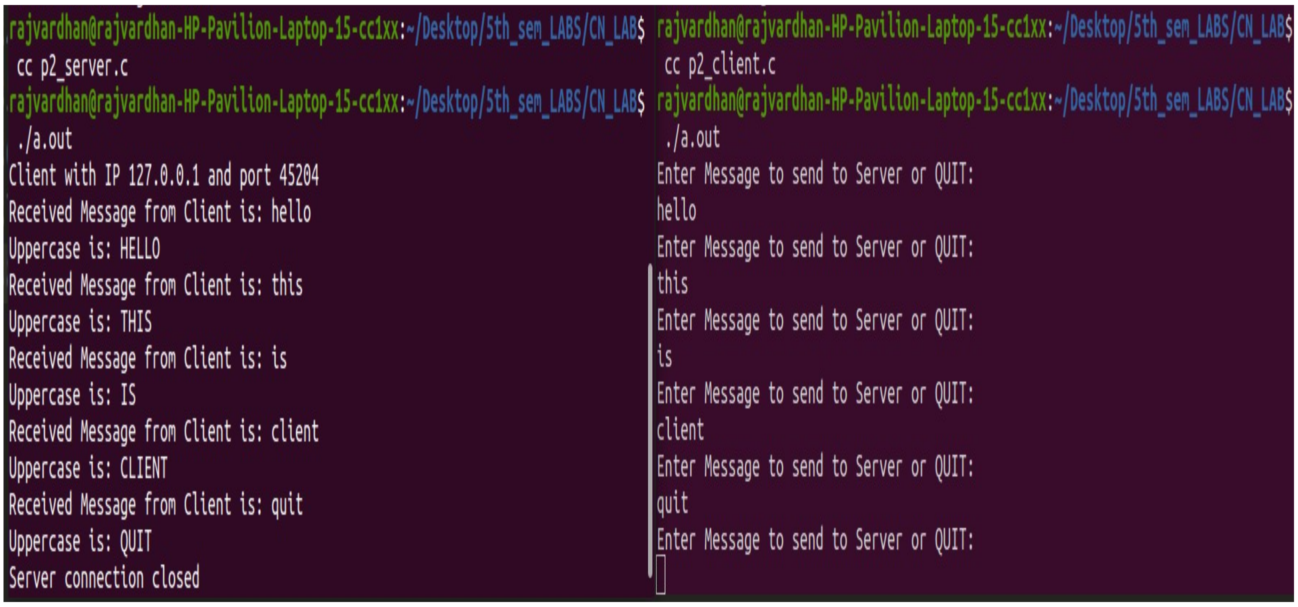
```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>
#include <ctype.h>
#define PORT 7000
#define sa struct sockaddr
int main()
```

```
{
 int sockid = socket(AF_INET, SOCK_STREAM, 0);
 int m = 0, n = 0, data_len, sockid_new;
 char buff[100];
 unsigned int len;
 struct sockaddr_in serv_addr, cli_addr;
 bzero(&serv_addr, sizeof(serv_addr));
 serv_addr.sin_family = AF_INET;
 serv_addr.sin_port = htons(PORT);
 serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
 if (bind(sockid, (sa *)&serv_addr, sizeof(serv_addr)) < 0)
 {
 printf("Could not bind socket");
 exit(0);
 }
 listen(sockid, 5);
 len = sizeof(cli_addr);
 sockid_new = accept(sockid, (sa *)&cli_addr, &len);
 printf("Client with IP %s and port %d\n", inet_ntoa(cli_addr.sin_addr), ntohs
(cli_addr.sin_port));
 for(;;)
 {
 bzero(buff, sizeof(buff));
 read(sockid_new, buff, sizeof(buff));
 printf("Received Message from Client is: %s\n", buff);
 for (int i = 0; i < strlen(buff); i++)
 buff[i] = toupper(buff[i]);
 // write(sockid_new, buff, sizeof(buff));
 printf("Uppercase is: %s\n", buff);
 if (strncmp(buff, "QUIT", 4) == 0)
 break;
 }
 printf("Server connection closed\n");
 close(sockid);
 return 0;
}
```

Client program :

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <string.h>
#define PORT 7000
#define sa struct sockaddr
int main()
{
 int sockid = socket(AF_INET, SOCK_STREAM, 0);
 int data_len;
 unsigned int len;
 struct sockaddr_in serv_addr, temp;
 bzero(&serv_addr, sizeof(serv_addr));
 serv_addr.sin_family = AF_INET;
 serv_addr.sin_port = htons(PORT);
 serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
 connect(sockid, (sa *)&serv_addr, sizeof(serv_addr));
 char buff[100], line[100];
 for(;;)
 {
 printf("Enter Message to send to Server or QUIT: \n");
 bzero(line, sizeof(line));
 bzero(buff, sizeof(buff));
 scanf("%s", line);
 write(sockid, line, strlen(line));
 // read(sockid, buff, sizeof(buff));
 // printf("\nServer says: %s\n", buff);
 if (strncmp(buff, "QUIT", 4) == 0)
 break;
 }
 printf("Client connection closed\n");
 close(sockid);
 return 0;
}
```

Output :



```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CN_LAB$
 cc p2_server.c
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CN_LAB$
 ./a.out
Client with IP 127.0.0.1 and port 45204
Received Message from Client is: hello
Uppercase is: HELLO
Received Message from Client is: this
Uppercase is: THIS
Received Message from Client is: is
Uppercase is: IS
Received Message from Client is: client
Uppercase is: CLIENT
Received Message from Client is: quit
Uppercase is: QUIT
Server connection closed
```

```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CN_LAB$
 cc p2_client.c
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CN_LAB$
 ./a.out
Enter Message to send to Server or QUIT:
hello
Enter Message to send to Server or QUIT:
this
Enter Message to send to Server or QUIT:
is
Enter Message to send to Server or QUIT:
client
Enter Message to send to Server or QUIT:
quit
Enter Message to send to Server or QUIT:
```