

COOCHBEHAR GOVT. ENGINEERING COLLEGE

NAME – SREEPARNA SINGHA

ROLL NO. – 34900319014

SUBJECT – COMPUTER NETWORK LAB ASSINGNMENT 2

DEPARTMENT – COMPUTER SCIENCE & ENGINEERING

SEMESTER – SIXTH

YEAR – THIRD

1. Write a client server program to communicate between them (two way).

Ans :

Client

```
#include<stdio.h>
#include<unistd.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<string.h>
int main()
{
    struct sockaddr_in c_addr;
    int c_fd,c_len;
    char buff[100];
    if((c_fd=socket(AF_INET,SOCK_STREAM,0))==-1)
        printf("[-]Error in Socket\n");
    printf("[+]Client Socket created\n");
    c_addr.sin_family=AF_INET;
    c_addr.sin_addr.s_addr=INADDR_ANY;
    c_addr.sin_port=3452;
    c_len=sizeof(c_addr);
    if(connect(c_fd,(struct sockaddr*)& c_addr,c_len)==-1)
        printf("[-]Error in Connect\n");
    printf("[+]Connected to the Server: \n\n");
    while(1)
    {
        read(c_fd,buff,100);
        printf("From Server: %s\n",buff);
        printf("Client Message: ");
        fgets(buff,sizeof(buff),stdin);
        if (strcmp(buff, "Exit\n") == 0){
            printf("[-]Disconneted from Server");
            write(c_fd,"Disconnected\n",100);
            break;
        }
        else{
```

```

write(c_fd, buff, 100);
}
}
close(c_fd);
return 0;
}

```

Server

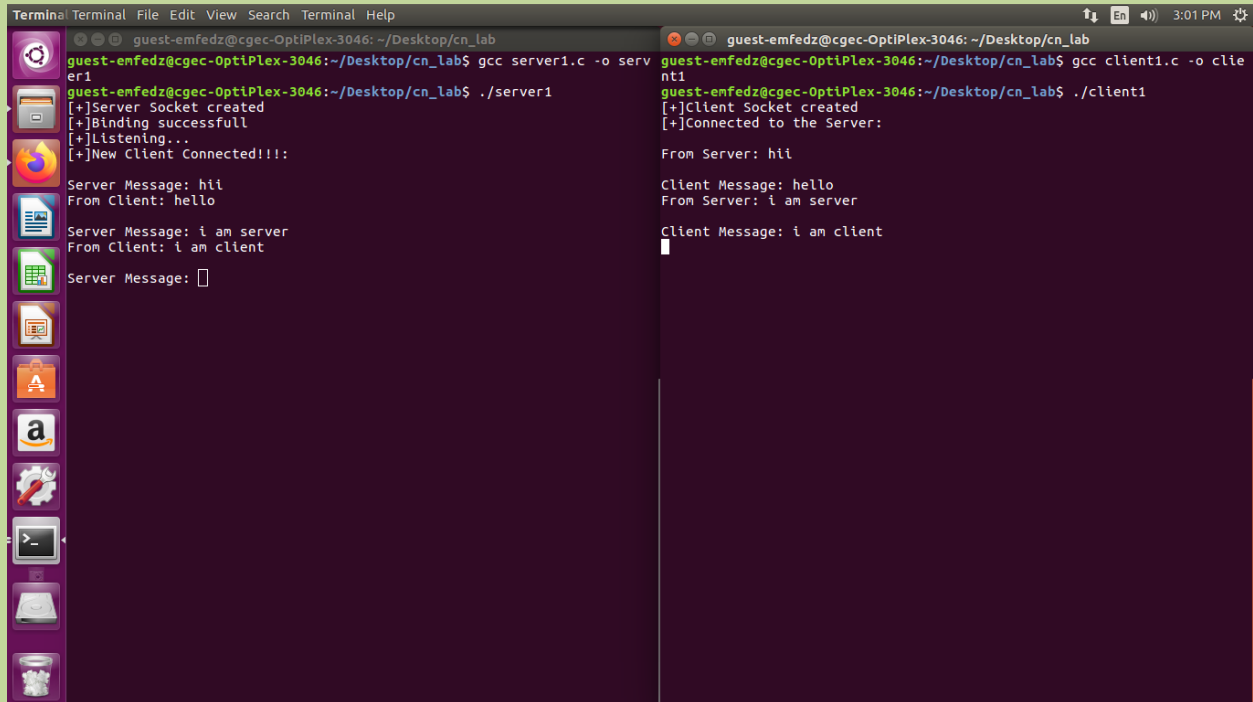
```

#include<stdio.h>
#include<unistd.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<sys/socket.h>
#include<sys/types.h>
int main()
{
    struct sockaddr_in s_addr, c_addr;
    int s_fd, c_fd, s_len, c_len;
    if((s_fd=socket(AF_INET, SOCK_STREAM, 0))==-1)
        printf("[-]Error in Socket\n");
    printf("[+]Server Socket created\n");
    s_addr.sin_family = AF_INET;
    s_addr.sin_port = 3452;
    s_len=sizeof(s_addr);
    if(bind(s_fd, (struct sockaddr*)& s_addr, s_len)==-1)
        printf("[-]Error in binding\n");
    printf("[+]Binding successfull\n");
    if(listen(s_fd, 5)==-1)
        printf("[-]Error in listen\n");
    printf("[+]Listening... \n");
    c_len=sizeof(c_addr);
    if((c_fd=accept(s_fd, (struct sockaddr*)&c_addr, &c_len))==-1)
        printf("\n[-]Error in accepting\n");
    printf("[+]New Client Connected!!!: \n\n");
    char buff[100];
    while(1)
    {
        printf("Server Message: ");
        fgets(buff, sizeof(buff), stdin);
        write(c_fd, buff, 100);
        read(c_fd, buff, 100);
        printf("From Client: %s\n", buff);
    }
    close(c_fd);
    return 0;
}

```

```
}
```

Output



The image shows two terminal windows side-by-side. The left window is the server, and the right window is the client. Both are running on a system with a purple desktop background and a sidebar of application icons.

```
Terminal Terminal File Edit View Search Terminal Help
guest-emfedz@cgec-OptiPlex-3046: ~/Desktop/cn_lab
guest-emfedz@cgec-OptiPlex-3046:~/Desktop/cn_lab$ gcc server1.c -o server1
guest-emfedz@cgec-OptiPlex-3046:~/Desktop/cn_lab$ ./server1
[+]Server Socket created
[+]Binding successfull
[+]Listening...
[+]New Client Connected!!!:
Server Message: hll
From Client: hello
Server Message: i am server
From Client: i am client
Server Message: 

```

```
guest-emfedz@cgec-OptiPlex-3046: ~/Desktop/cn_lab
guest-emfedz@cgec-OptiPlex-3046:~/Desktop/cn_lab$ gcc client1.c -o client1
guest-emfedz@cgec-OptiPlex-3046:~/Desktop/cn_lab$ ./client1
[+]Client Socket created
[+]Connected to the Server:
From Server: hll
Client Message: hello
From Server: i am server
Client Message: i am client

```

2. Write a program that will ask the client program to enter two numbers and the server will display the addition, subtraction, and multiplication of that two numbers.

Ans:

Client

```
#include <stdio.h>
#include <unistd.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <string.h>
#include <stdlib.h>
int main()
{
    struct sockaddr_in s_addr, c_addr;
    int s_fd, c_fd, s_len, c_len;
    if ((s_fd = socket(AF_INET, SOCK_STREAM, 0)) == -1)
        printf("[-]Error in Socket\n");
    printf("[+]Server Socket created\n");
    s_addr.sin_family = AF_INET;
    s_addr.sin_port = 3452;
    s_len = sizeof(s_addr);
    if (bind(s_fd, (struct sockaddr *)&s_addr, s_len) == -1)
        printf("[-]Error in binding\n");
    printf("[+]Binding successfull\n");
    if (listen(s_fd, 5) == -1)
        printf("[-]Error in listen\n");
    printf("[+]Listening... \n");
    int nums[2], n = 0;
    c_len = sizeof(c_addr);
    if ((c_fd = accept(s_fd, (struct sockaddr *)&c_addr, &c_len))
        == -1)
        printf("\n[-]Error in accepting\n");
    printf("[+]New Client Connected!!!: \n\n");
    while (n < 2)
    {
        char buff[100];
        read(c_fd, buff, 100);
```

```

printf("From Client Number %d is %s", n + 1, buff);
nums[n++] = atoi(buff);
}
printf("\nAddition of %d and %d is %d\n", nums[0], nums[1],
nums[0] + nums[1]);
printf("Subtaction of %d and %d is %d\n", nums[0], nums[1],
nums[0] - nums[1]);
printf("Multiplication of %d and %d is %d\n", nums[0],
nums[1], nums[0] * nums[1]);
close(c_fd);
return 0;
}

```

Server

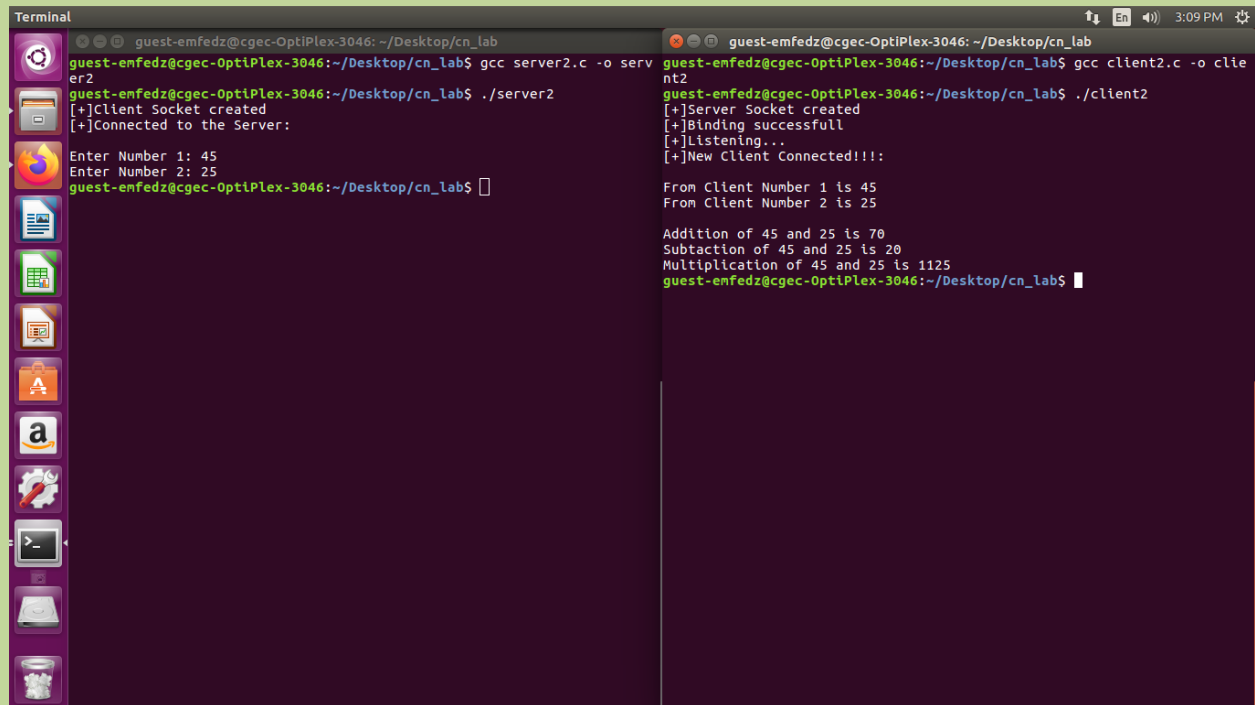
```

#include <stdio.h>
#include <unistd.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <string.h>
#include <stdlib.h>
int main()
{
struct sockaddr_in c_addr;
int c_fd, c_len;
char buff[100];
if ((c_fd = socket(AF_INET, SOCK_STREAM, 0)) == -1)
printf("[-]Error in Socket\n");
printf("[+]Client Socket created\n");
c_addr.sin_family = AF_INET;
c_addr.sin_addr.s_addr = INADDR_ANY;
c_addr.sin_port = 3452;
c_len = sizeof(c_addr);
if (connect(c_fd, (struct sockaddr *)&c_addr, c_len) == -1)
printf("[-]Error in Connect\n");
printf("[+]Connected to the Server: \n\n");
int n = 0;
while (n < 2)
{
printf("Enter Number %d: ", n + 1);
fgets(buff, sizeof(buff), stdin);
write(c_fd, buff, 100);
n++;
}
}

```

```
close(c_fd);  
return 0;  
}
```

Output



The image shows two terminal windows side-by-side. The left window is titled 'Terminal' and shows the execution of a server program. The user enters 'gcc server2.c -o server2' and then './server2'. The program outputs '[+]Client Socket created' and '[+]Connected to the Server:'. It then prompts for two numbers: 'Enter Number 1: 45' and 'Enter Number 2: 25'. The right window is titled 'Terminal' and shows the execution of a client program. The user enters 'gcc client2.c -o client2' and then './client2'. The program outputs '[+]Server Socket created', '[+]Binding successfull', '[+]Listening...', and '[+]New Client Connected!!!'. It then displays the results of calculations: 'From Client Number 1 is 45', 'From Client Number 2 is 25', 'Addition of 45 and 25 is 70', 'Subtraction of 45 and 25 is 20', and 'Multiplication of 45 and 25 is 1125'.

```
Terminal guest-emfedz@cgec-OptiPlex-3046: ~/Desktop/cn_lab  
guest-emfedz@cgec-OptiPlex-3046:~/Desktop/cn_lab$ gcc server2.c -o server2  
guest-emfedz@cgec-OptiPlex-3046:~/Desktop/cn_lab$ ./server2  
[+]Client Socket created  
[+]Connected to the Server:  
Enter Number 1: 45  
Enter Number 2: 25  
guest-emfedz@cgec-OptiPlex-3046:~/Desktop/cn_lab$  
  
Terminal guest-emfedz@cgec-OptiPlex-3046: ~/Desktop/cn_lab  
guest-emfedz@cgec-OptiPlex-3046:~/Desktop/cn_lab$ gcc client2.c -o client2  
guest-emfedz@cgec-OptiPlex-3046:~/Desktop/cn_lab$ ./client2  
[+]Server Socket created  
[+]Binding successfull  
[+]Listening...  
[+]New Client Connected!!!  
From Client Number 1 is 45  
From Client Number 2 is 25  
Addition of 45 and 25 is 70  
Subtraction of 45 and 25 is 20  
Multiplication of 45 and 25 is 1125  
guest-emfedz@cgec-OptiPlex-3046:~/Desktop/cn_lab$
```

3. Write a program to transfer a text file from the client to server and vice-versa.

Ans:

Client

```
#include <arpa/inet.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <unistd.h>
#define IP_PROTOCOL 0
#define IP_ADDRESS "127.0.0.1" // localhost
#define PORT_NO 15050
#define NET_BUF_SIZE 32
#define cipherKey 'S'
#define sendrecvflag 0
// function to clear buffer
void clearBuf(char* b)
{
    int i;
    for (i = 0; i < NET_BUF_SIZE; i++)
        b[i] = '\0';
}
// function for decryption
char Cipher(char ch)
{
    return ch ^ cipherKey;
}
// function to receive file
int recvFile(char* buf, int s)
{
    int i;
    char ch;
    for (i = 0; i < s; i++) {
        ch = buf[i];
        ch = Cipher(ch);
        if (ch == EOF)
            return 1;
    }
}
```



```

else
printf("%c", ch);
}
return 0;
}
// driver code
int main()
{
int sockfd, nBytes;
struct sockaddr_in addr_con;
int addrlen = sizeof(addr_con);
addr_con.sin_family = AF_INET;
addr_con.sin_port = htons(PORT_NO);
addr_con.sin_addr.s_addr = inet_addr(IP_ADDRESS);
char net_buf[NET_BUF_SIZE];
FILE* fp;
// socket()
sockfd = socket(AF_INET, SOCK_DGRAM,
IP_PROTOCOL);
if (sockfd < 0)
printf("\nfile descriptor not received!!\n");
else
printf("\nfile descriptor %d received\n", sockfd);
while (1) {
printf("\nPlease enter file name to receive:\n");
scanf("%s", net_buf);
sendto(sockfd, net_buf, NET_BUF_SIZE,
sendrecvflag, (struct sockaddr*)&addr_con,
addrlen);
printf("\n-----Data Received-----\n");
while (1) {
// receive
clearBuf(net_buf);
nBytes = recvfrom(sockfd, net_buf, NET_BUF_SIZE,
sendrecvflag, (struct sockaddr*)&addr_con,
&addrlen);
// process
if (recvFile(net_buf, NET_BUF_SIZE)) {
break;
}
}
printf("\n-----\n");
}
return 0;
}

```

Server

```
#include <arpa/inet.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <unistd.h>
#define IP_PROTOCOL 0
#define PORT_NO 15050
#define NET_BUF_SIZE 32
#define cipherKey 'S'
#define sendrecvflag 0
#define nofile "File Not Found!"
// function to clear buffer
void clearBuf(char* b)
{
    int i;
    for (i = 0; i < NET_BUF_SIZE; i++)
        b[i] = '\0';
}
// function to encrypt
char Cipher(char ch)
{
    return ch ^ cipherKey;
}
// function sending file
int sendFile(FILE* fp, char* buf, int s)
{
    int i, len;
    if (fp == NULL) {
        strcpy(buf, nofile);
        len = strlen(nofile);
        buf[len] = EOF;
        for (i = 0; i <= len; i++)
            buf[i] = Cipher(buf[i]);
        return 1;
    }
    char ch, ch2;
    for (i = 0; i < s; i++) {
        ch = fgetc(fp);
        ch2 = Cipher(ch);
        buf[i] = ch2;
    }
}
```

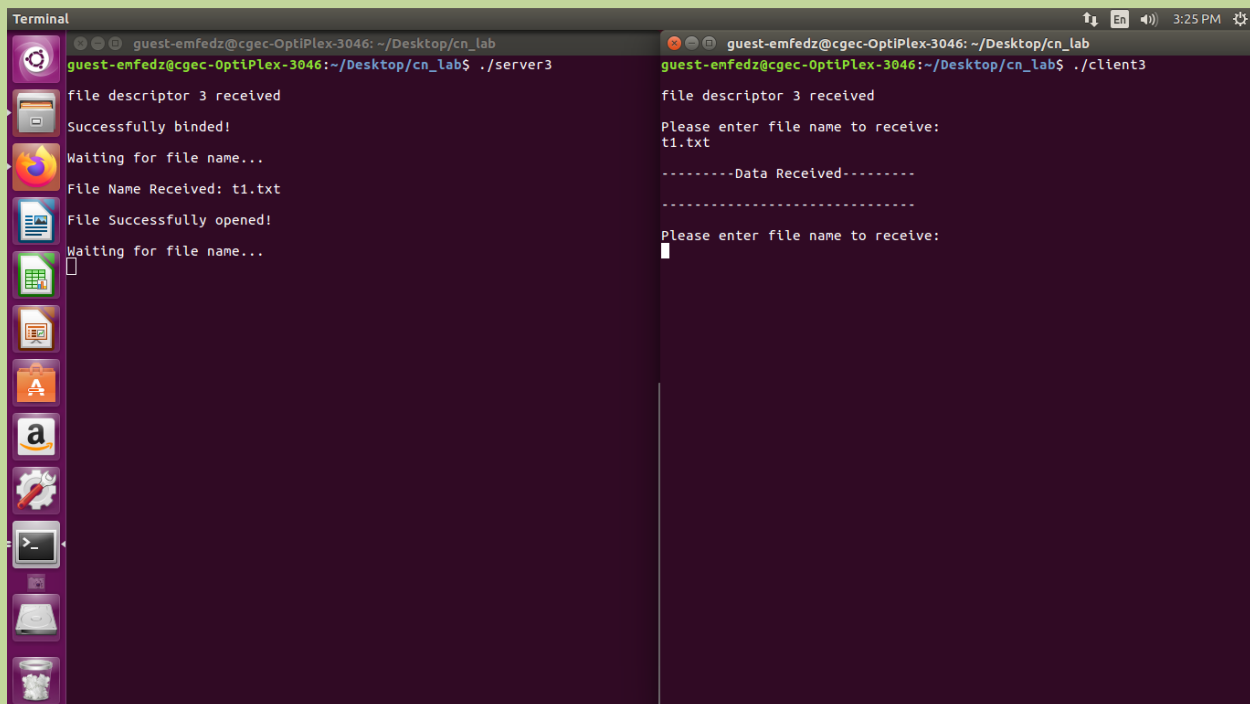
```

if (ch == EOF)
return 1;
}
return 0;
}
// driver code
int main()
{
int sockfd, nBytes;
struct sockaddr_in addr_con;
int addrlen = sizeof(addr_con);
addr_con.sin_family = AF_INET;
addr_con.sin_port = htons(PORT_NO);
addr_con.sin_addr.s_addr = INADDR_ANY;
char net_buf[NET_BUF_SIZE];
FILE* fp;
// socket()
sockfd = socket(AF_INET, SOCK_DGRAM, IP_PROTOCOL);
if (sockfd < 0)
printf("\nfile descriptor not received!!\n");
else
printf("\nfile descriptor %d received\n", sockfd);
// bind()
if (bind(sockfd, (struct sockaddr*)&addr_con, sizeof(addr_con))
== 0)
printf("\nSuccessfully binded!\n");
else
printf("\nBinding Failed!\n");
while (1) {
printf("\nWaiting for file name...\n");
// receive file name
clearBuf(net_buf);
nBytes = recvfrom(sockfd, net_buf,
NET_BUF_SIZE, 0,
(struct sockaddr*)&addr_con, &addrlen);
fp = fopen(net_buf, "r");
printf("\nFile Name Received: %s\n", net_buf);
if (fp == NULL)
printf("\nFile open failed!\n");
else
printf("\nFile Successfully opened!\n");
while (1) {
// process
if (sendFile(fp, net_buf, NET_BUF_SIZE)) {
sendto(sockfd, net_buf, NET_BUF_SIZE,

```

```
sendrecvflag,  
(struct sockaddr*)&addr_con, addrlen);  
break;  
}  
// send  
sendto(sockfd, net_buf, NET_BUF_SIZE,  
sendrecvflag,  
(struct sockaddr*)&addr_con, addrlen);  
clearBuf(net_buf);  
}  
if (fp != NULL)  
fclose(fp);  
}  
return 0;  
}
```

Output



The screenshot displays two terminal windows side-by-side. The left window, titled 'Terminal', shows the execution of a server program. The user runs './server3' in the directory ~/Desktop/cn_lab. The output shows: 'file descriptor 3 received', 'Successfully binded!', 'Waiting for file name...', 'File Name Received: t1.txt', 'File Successfully opened!', and 'Waiting for file name...'. The right window, also titled 'Terminal', shows the execution of a client program. The user runs './client3' in the same directory. The output shows: 'file descriptor 3 received', 'Please enter file name to receive: t1.txt', '-----Data Received-----', and 'Please enter file name to receive:'. The system clock in the top right corner indicates 3:25 PM.

```
Terminal
guest-emfedz@cgec-OptiPlex-3046: ~/Desktop/cn_lab
guest-emfedz@cgec-OptiPlex-3046:~/Desktop/cn_lab$ ./server3
file descriptor 3 received
Successfully binded!
Waiting for file name...
File Name Received: t1.txt
File Successfully opened!
Waiting for file name...

guest-emfedz@cgec-OptiPlex-3046: ~/Desktop/cn_lab
guest-emfedz@cgec-OptiPlex-3046:~/Desktop/cn_lab$ ./client3
file descriptor 3 received
Please enter file name to receive:
t1.txt
-----Data Received-----
Please enter file name to receive:
```

4. A database is created with the following fields: roll no, student name, address, marks1, marks2. The database will be stored in the server and the client will fetch the information of a student by sending the roll no of a particular student. Implement this scenario using client server program.

Ans:

Client

```
#include<stdio.h>
#include<unistd.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<string.h>
#include<stdlib.h>
int main()
{
    struct sockaddr_in c_addr;
    int c_fd,c_len;
    char buff[100];
    if((c_fd=socket(AF_INET,SOCK_STREAM,0))== -1){
        printf("[ -]Error in Socket\n");
        exit(0);
    }
    printf("[+]Client Socket created\n");
    c_addr.sin_family=AF_INET;
    c_addr.sin_addr.s_addr=INADDR_ANY;
    c_addr.sin_port=3452;
    c_len=sizeof(c_addr);
    if(connect(c_fd,(struct sockaddr*)& c_addr,c_len)== -1) {
        printf("[ -]Error in Connect\n");
        exit(0);
    }
    printf("[+]Connected to the Server: \n\n");
    while(1)
    {
        printf("\nEnter Roll No of Student: \n");
        fgets(buff,sizeof(buff),stdin);
```

```

if (strcmp(buff, "Exit\n") == 0){
printf("[-]Disconnected from Server");
write(c_fd,"Disconnected\n",100);
break;
}
else{
write(c_fd,buff,100);
}
read(c_fd,buff,100);
printf("%s",buff);
}
close(c_fd);
return 0;
}

```

Server

```

#include<stdio.h>
#include<unistd.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<stdlib.h>
struct StudentDetails
{
int roll_no;
char* name;
char* address;
float marks1;
float marks2;
};
int main()
{
struct sockaddr_in s_addr,c_addr;
int s_fd,c_fd,s_len,c_len;
struct StudentDetails studentdet[13];
studentdet[0].roll_no = 1;
studentdet[0].name = "Indraneel";
studentdet[0].address = "CGEC";
studentdet[0].marks1 = 89.0;
studentdet[0].marks2 = 99.0;
studentdet[1].roll_no = 2;
studentdet[1].name = "Sachin";
studentdet[1].address = "CGEC";
studentdet[1].marks1 = 81.0;

```

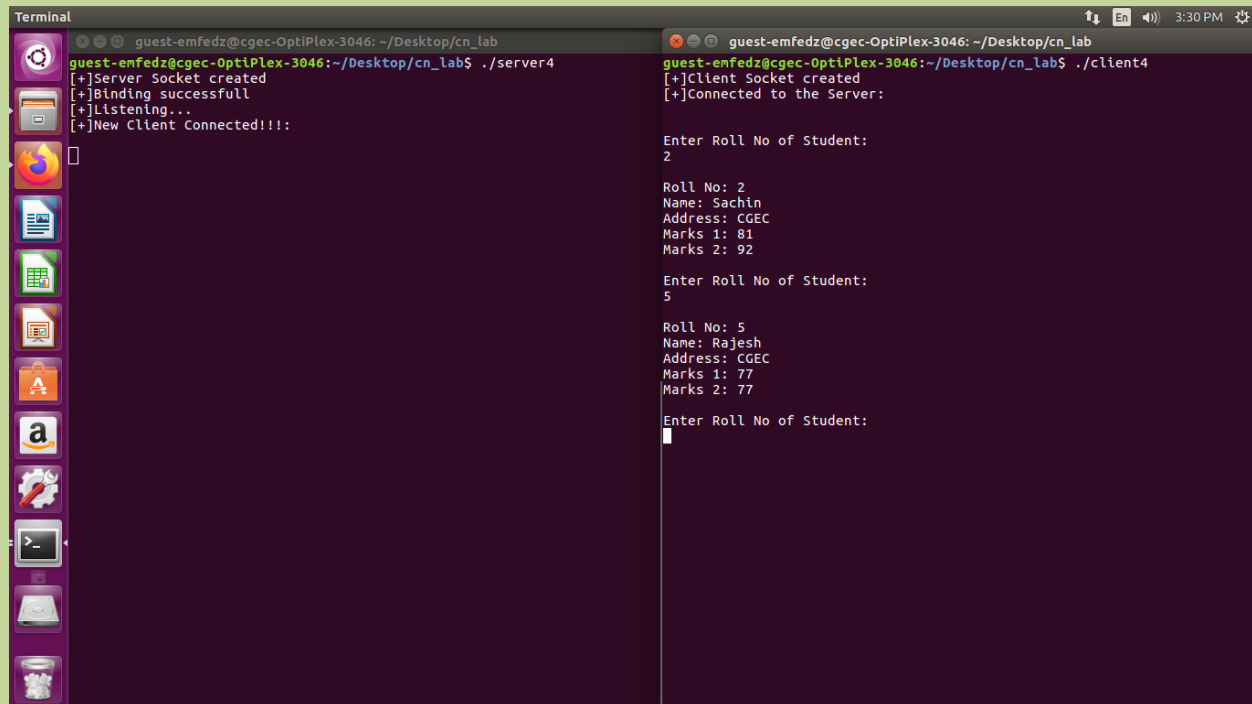
```
studentdet[1].marks2 = 92.0;
studentdet[2].roll_no = 3;
studentdet[2].name = "Saurabh";
studentdet[2].address = "CGEC";
studentdet[2].marks1 = 76.0;
studentdet[2].marks2 = 87.0;
studentdet[3].roll_no = 4;
studentdet[3].name = "Indra";
studentdet[3].address = "CGEC";
studentdet[3].marks1 = 87.0;
studentdet[3].marks2 = 94.0;
studentdet[4].roll_no = 5;
studentdet[4].name = "Rajesh";
studentdet[4].address = "CGEC";
studentdet[4].marks1 = 77.0;
studentdet[4].marks2 = 77.0;
studentdet[5].roll_no = 6;
studentdet[5].name = "king";
studentdet[5].address = "CGEC";
studentdet[5].marks1 = 89.0;
studentdet[5].marks2 = 91.0;
studentdet[6].roll_no = 7;
studentdet[6].name = "Raj";
studentdet[6].address = "CGEC";
studentdet[6].marks1 = 81.0;
studentdet[6].marks2 = 91.0;
studentdet[7].roll_no = 8;
studentdet[7].name = "Ravi";
studentdet[7].address = "CGEC";
studentdet[7].marks1 = 86.0;
studentdet[7].marks2 = 90.0;
studentdet[8].roll_no = 9;
studentdet[8].name = "sourav";
studentdet[8].address = "CGEC";
studentdet[8].marks1 = 90.0;
studentdet[8].marks2 = 90.0;
studentdet[9].roll_no = 10;
studentdet[9].name = "Neel";
studentdet[9].address = "CGEC";
studentdet[9].marks1 = 89.0;
studentdet[9].marks2 = 92.0;
if((s_fd=socket(AF_INET, SOCK_STREAM, 0))==-1){
printf("[-]Error in Socket\n");
exit(0);
}
```

```

printf("[+]Server Socket created\n");
s_addr.sin_family = AF_INET;
s_addr.sin_port = 3452;
s_len=sizeof(s_addr);
if(bind(s_fd,(struct sockaddr*)& s_addr,s_len)==-1){
printf("[-]Error in binding\n");
exit(0);
}
printf("[+]Binding successfull\n");
if(listen(s_fd,5)==-1){
printf("[-]Error in listen\n");
exit(0);
}
printf("[+]Listening... \n");
c_len=sizeof(c_addr);
if((c_fd=accept(s_fd,(struct sockaddr*)&c_addr,&c_len))==-1){
printf("\n[-]Error in accepting\n");
exit(0);
}
printf("[+]New Client Connected!!!: \n\n");
char buff[100];
while(1)
{
int roll;
read(c_fd,buff,100);
roll = atoi(buff);
for(int i = 0; i < 10; i++){
if(studentdet[i].roll_no == roll){
snprintf(buff, 100, "\nRoll No: %d\nName:
%s\nAddress: %s\nMarks 1: %2.f\nMarks 2: %2.f\n"
,studentdet[i].roll_no,
studentdet[i].name,studentdet[i].address,
studentdet[i].marks1, studentdet[i].marks2);
write(c_fd,buff,100);
printf("[+]Detail found and sent to client successfully!");
break;}
else if(studentdet[i].roll_no != roll && i == 9){
printf("[-]Details not found.");
write(c_fd,"\n[-]Details not found.\n",100);
}
}
}
close(c_fd);
return 0;
}

```


Output



```
Terminal
guest-emfedz@cgec-OptiPlex-3046: ~/Desktop/cn_lab
guest-emfedz@cgec-OptiPlex-3046:~/Desktop/cn_lab$ ./server4
[+]Server Socket created
[+]Binding successfull
[+]Listening...
[+]New Client Connected!!!

guest-emfedz@cgec-OptiPlex-3046: ~/Desktop/cn_lab
guest-emfedz@cgec-OptiPlex-3046:~/Desktop/cn_lab$ ./client4
[+]Client Socket created
[+]Connected to the Server:

Enter Roll No of Student:
2

Roll No: 2
Name: Sachin
Address: CGEC
Marks 1: 81
Marks 2: 92

Enter Roll No of Student:
5

Roll No: 5
Name: Rajesh
Address: CGEC
Marks 1: 77
Marks 2: 77

Enter Roll No of Student:

```

5. Write a program to implement Stop & Wait ARQ method.

Ans:

Client

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#define MAXLINE 80
char buffer[255] ;
#define SERVER_ADDR "127.0.0.1"
#define SERVER_PORT_NUM 6000
#define t_max_retransmission 3
typedef int bit_32_var ;
typedef char bit_8_var ;
typedef struct pdu_field
{
    bit_32_var SN ;
    bit_8_var data[MAXLINE] ;
    bit_8_var status ;
}PDU_FIELD;
void str_cli ( FILE *fp , bit_32_var sfd )
{
    bit_32_var no_of_data , counter_1 , counter_2 ;
    static bit_32_var retransmission_counter ,
    initial_readycheck_counter ;
    PDU_FIELD send_data[MAXLINE] ,recv_data[MAXLINE] ;
    printf ( "enter how many data you have to send : " ) ;
    scanf ( "%d", &no_of_data ) ;
    for ( counter_1 = 0 ; counter_1 < no_of_data ; counter_1++ )
    {
        printf ( "enter %d'th data : ",counter_1 ) ;
        scanf ( "%s", send_data[counter_1].data ) ;
        send_data[counter_1].SN = counter_1 ;
        send_data[counter_1].status = 0 ;
    }
    for ( counter_1 = 0 ; counter_1 < no_of_data ; counter_1++ )
    {
```

```

write ( sfd , &send_data[counter_1] , sizeof (
send_data[counter_1] ) ) ;
read ( sfd , &recv_data[counter_1] , sizeof ( recv_data[counter_1]
) ) ;
if ( counter_1 == 0 && ( strcmp ( recv_data[counter_1].data ,
"yes" ) !=
0 ) )
{
if ( intial_readycheck_counter == t_max_retransmission )
{
printf ( "client : receiver not there , better to exit : \n" ) ;
exit ( 1 ) ;
}
intial_readycheck_counter++ ;
printf ( "client : receiver is not ready : wait 10sec... \n" ) ;
sleep ( 10 ) ;
counter_1 = counter_1- 1 ;
}
if ( counter_1 > 0 )
{
if ( recv_data[counter_1].SN == send_data[counter_1].SN + 1 )
printf ( "\n server responding = %s ", recv_data[counter_1].data )
;
else
{
if ( retransmission_counter < t_max_retransmission )
{
printf ( "incorrect ack - sending the same data - \n" ) ;
counter_1 -= 1 ;
retransmission_counter++ ;
}
else
{
printf ( "Time out : sending next data \n " ) ;
retransmission_counter = 0 ;
}
}
}
}
}
}
write ( sfd , "Nothingtotransmit:\n" , 10 ) ;
printf ( "Transmission finished : \n" ) ;
exit ( 1 ) ;
}
int main ()
{

```

```

bit_32_var socket_fd ;
struct sockaddr_in config_client ;
socket_fd = socket ( AF_INET , SOCK_STREAM , 0 ) ;
if ( socket_fd < 0 )
{
printf ( "client : failed to create socket \n" ) ;
exit ( 1 ) ;
}
memset ( &config_client , 0 , sizeof ( struct sockaddr_in ) ) ;
config_client.sin_family = AF_INET ;
config_client.sin_port = htons ( SERVER_PORT_NUM ) ;
inet_aton ( SERVER_ADDR , &config_client.sin_addr ) ;
connect ( socket_fd , (struct sockaddr *)&config_client , sizeof (
config_client ) ) ;
printf ( "connect successfully\n" ) ;
system ( "clear" ) ;
str_cli ( stdin , socket_fd ) ;
exit ( 0 ) ;
}

```

Server

```

#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h>
#define SUCCESS 1
#define FAILURE 0
#define NO_OF_FRAMES 10
typedef int BIT_32_VAR_INT ;
typedef char BIT_8_VAR_CHAR ;
void error ( BIT_8_VAR_CHAR *msg )
{
perror ( msg ) ;
exit ( 1 ) ;
}
typedef struct pdu_data
{
BIT_32_VAR_INT SN ;
BIT_8_VAR_CHAR data[80] ;
BIT_32_VAR_INT status ;
}PDU_DATA;

```

```

int main () {
PDU_DATA send_data[10] , recv_data[10] ;
BIT_32_VAR_INT socket_fd , newsocket_fd , port_no , client_len ;
// char buffer[256] ;
struct sockaddr_in serv_addr , cli_addr ;
int sizeof_read_data , counter_1 ;
//char buf[256] ;
socket_fd = socket ( AF_INET , SOCK_STREAM , 0 ) ;
if ( socket_fd < 0 )
{
error ( "error in socket opening " ) ;
}
bzero ( ( char * ) &serv_addr , sizeof ( serv_addr ) ) ;
port_no = atoi ( "7100" ) ;
serv_addr.sin_family = AF_INET ;
serv_addr.sin_addr.s_addr = INADDR_ANY ;
serv_addr.sin_port = htons ( 6000 ) ;
if ( bind ( socket_fd , ( struct sockaddr * ) &serv_addr , sizeof
(
serv_addr ) ) < 0 )
error ( "error in binding" ) ;
listen ( socket_fd , 5 ) ;
client_len = sizeof ( cli_addr ) ;
newsocket_fd = accept ( socket_fd , ( struct sockaddr * )
&cli_addr ,
&client_len ) ;
if ( newsocket_fd < 0 )
error ( "error in accept " ) ;
// bzero ( buffer , 256 ) ;
while ( 1 )
{
for ( counter_1 = 0 ; counter_1 < NO_OF_FRAMES ; counter_1++ )
{
sizeof_read_data = read ( newsocket_fd , &recv_data[counter_1] ,
sizeof ( recv_data[counter_1]) ) ;
if ( sizeof_read_data < 0 )
error ( "error in reading from socket " ) ;
printf ( "client sended : %s with SN %d :\n" ,
recv_data[counter_1].data , recv_data[counter_1].SN ) ;
printf ( "enter a reply to client : " ) ;
scanf ( "%s",send_data[counter_1].data ) ;
printf ( "\nenter a SN for next you want to receive : " ) ;
scanf ( "%d",&send_data[counter_1].SN ) ;
if ( send_data[counter_1].SN <= recv_data[counter_1].SN )
counter_1 -= 1 ;
}
}
}

```

```

send_data[counter_1].status = SUCCESS ;
write ( newsocket_fd , &send_data[counter_1] , sizeof (
send_data[counter_1]) ) ;
}
bzero ( &recv_data[counter_1].data , sizeof (
recv_data[counter_1].data
) ) ;
read ( newsocket_fd , &recv_data[counter_1] , sizeof (
recv_data[counter_1] ) ) ;
printf ( "it's end Thank YOu : \n" ) ;
exit ( 1 ) ;
}
return 0 ;
}

```

Output

```

Terminal
guest-emfedz@cgec-OptiPlex-3046: ~/Desktop/cn_lab
guest-emfedz@cgec-OptiPlex-3046:~/Desktop/cn_lab$ gcc server5.c -o serv
er5
server5.c: In function 'main':
server5.c:65:23: warning: implicit declaration of function 'read' [-Wim
plicit-function-declaration]
    sizeof_read_data = read ( newsocket_fd , &recv_data[counter_1] , st
    ^
server5.c:82:4: warning: implicit declaration of function 'write' [-Wim
plicit-function-declaration]
    write ( newsocket_fd , &send_data[counter_1] , sizeof ( send_data[
    ^
guest-emfedz@cgec-OptiPlex-3046:~/Desktop/cn_lab$ ./server5
client sended : 2 with SN 0 :
enter a reply to client : 3

enter a SN for next you want to receive : 3
client sended : 2 with SN 0 :
enter a reply to client : 2

enter a SN for next you want to receive : 2
client sended : 2 with SN 0 :
enter a reply to client : 6

enter a SN for next you want to receive : 1
client sended : 2 with SN 0 :
enter a reply to client : 8

enter a SN for next you want to receive : 6
client sended : with SN 0 :
enter a reply to client :

enter how many data you have to send : 3
enter 0'th data : 2
enter 1'th data : 4
enter 2'th data : 6
client : receiver is not ready : wait 10sec...
client : receiver is not ready : wait 10sec...
client : receiver is not ready : wait 10sec...
client : receiver not there , better to exit :
guest-emfedz@cgec-OptiPlex-3046:~/Desktop/cn_lab$

```

6. Write a program to implement Go-Back-N ARQ method.

Ans:

```
#include<bits/stdc++.h>
#include<ctime>
#define ll long long int
using namespace std;
void transmission(ll & i, ll & N, ll & tf, ll & tt) {
while (i <= tf) {
int z = 0;
for (int k = i; k < i + N && k <= tf; k++) {
cout << "Sending Frame " << k << "..." << endl;
tt++;
}
for (int k = i; k < i + N && k <= tf; k++) {
int f = rand() % 2;
if (!f) {
cout << "Acknowledgment for Frame " << k << "..." <<
endl;
z++;
} else {
cout << "Timeout!! Frame Number : " << k << " Not
Received" << endl;
cout << "Retransmitting Window..." << endl;
break;
}
}
}
cout << "\n";
i = i + z;
}
}
int main() {
ll tf, N, tt = 0;
srand(time(NULL));
cout << "Enter the Total number of frames : ";
cin >> tf;
cout << "Enter the Window Size : ";
cin >> N;
ll i = 1;
transmission(i, N, tf, tt);
cout << "Total number of frames which were sent and resent are
: " << tt <<
endl;
return 0;
}
```

```
}
```

Output

```
guest-emfedz@cgcec-OptiPlex-3046:~/Desktop/cn_lab$ g++ 6.c -o 6
guest-emfedz@cgcec-OptiPlex-3046:~/Desktop/cn_lab$ ./6
Enter the Total number of frames : 3
Enter the Window Size : 2
Sending Frame 1...
Sending Frame 2...
Acknowledgment for Frame 1...
Timeout!! Frame Number : 2 Not Received
Retransmitting Window...
Sending Frame 2...
Sending Frame 3...
Timeout!! Frame Number : 2 Not Received
Retransmitting Window...
Sending Frame 2...
Sending Frame 3...
Timeout!! Frame Number : 2 Not Received
Retransmitting Window...
Sending Frame 2...
Sending Frame 3...
Acknowledgment for Frame 2...
Acknowledgment for Frame 3...
Total number of frames which were sent and resent are : 8
guest-emfedz@cgcec-OptiPlex-3046:~/Desktop/cn_lab$
```


7. Write a program to implement Selective Repeat ARQ method.

Ans:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <unistd.h>
int n,r;
struct frame{
char ack;
int data;
}frm[10];
int sender(void);
void rcvack(void);
void resend_sr(void);
void resend_gb(void);
void goback(void);
void selective(void);
int main(){
int c;
do{
printf("1:Selective Repeat protocol\n2:Go Back N
protocol\n0:Exit\n");
printf("Enter choice:");
scanf("%d",&c);
switch(c)
{
case 1:selective();
break;
case 2:goback();
case 0: exit(0);
break;
}
}while(c>=1);
} void goback(){
sender();
rcvack();
resend_gb();
printf("\nAll frames sent successfully!\n");
}
void selective(){
sender();
rcvack();
```

```

resend_sr();
printf("\nAll frames sent successfully!\n");
}
int sender(){
int i;
printf("\nEnter no. of frame to be sent:");
scanf("%d",&n);
for(i=1;i<=n;i++){
printf("\n Enter data for frame [%d] ",i);
scanf("%d",&frm[i].data);
frm[i].ack='y';
}
return 0;
}
void recvack(){
int i;
rand();
r=rand()%n;
frm[r].ack='n';
for(i=1;i<=n;i++){
if(frm[i].ack=='n'){
printf("\n The frame number %d is not
received\n",r);}
}
}
void resend_sr(){
printf("\nResending the frame %d",r);
sleep(2);
frm[r].ack='y';
printf("\nThe received frame is %d",frm[r].data);
}
void resend_gb(){
int i;
printf("\nResending from frame %d",r);
for(i=r;i<=n;i++){
sleep(2);
frm[i].ack='y';
printf("\nReceived data of frame %d is
%d",i,frm[i].data);
}
}
}

```

Output

```
guest-emfedz@cgec-OptiPlex-3046: ~/Desktop/cn_lab
guest-emfedz@cgec-OptiPlex-3046:~/Desktop/cn_lab$ gcc 7.c -o 7
guest-emfedz@cgec-OptiPlex-3046:~/Desktop/cn_lab$ ./7
1:Selective Repeat protocol
2:Go Back N protocol
0:Exit
Enter choice:2
Enter no. of frame to be sent:3
Enter data for frame [1] 2
Enter data for frame [2] 25
Enter data for frame [3] 53
The frame number 1 is not received
Resending from frame 1
Received data of frame 1 is 2
Received data of frame 2 is 25
Received data of frame 3 is 53
All frames sent successfully!
guest-emfedz@cgec-OptiPlex-3046:~/Desktop/cn_lab$
```