Output:

```
$ ./server

  Listening

Client 2 disconnected!
```

Client 1

```
$ ./client

 Hello
 Recieved : Hello
  Hi
 Recieved : Good morning
  Do yo have some work now?

 Recieved, Yes, Will talk later
   Bye then
   Recived : Bye
$ ^c
```

Client 2

```
$ ./client

 Hello
 Recieved : Hi
 Good morning
Recieved : Doyou have
 Some work now?
 Yes , will talk later
 Recieved : Bye then
   Bye
 ^c
$
```

5. Write a program to implement concurrent chat server that allows current logged-in users to comunicate with the other.

```
// Server

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

# define BUFSIZE 1024
# define PORT 5000
# define IP INADDR_ANY
# define BACKLOG 5

void servfunc ( int sockfd1 , int sockfd2)
{
        char buf [BUFSIZE];
        int cnt;
        cnt = recv( sockfd 2, buf, BUFSIZE, 0);
        if ( cnt==0)
        {    printf(" Client 2 disconnected"); exit(0);}
```

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date ______________ Name M. C. SOHAN

Dept./Lab ______________ Class ________ Expt./No. 5

Title ______________

```
write(1, buf, But);
 printf("2\n");
  bzero( buf, BUFSIZE);
  while( cnt= recv (sockfd1, buf, BUFSIZE,0))
  {
        send( sockfd2 , buf,cnt,0) ;
        write(1, buf, cnt);
         printf("1\n");
        bzero ( buf, BUFSIZE);
        if (cnt = read ( sockfd2 , buf , BUFSIZE))
                  send ( sockfd1, buf, cnt, 0);
         else {
              printf( "client 2 disconnected");
               close (sockfd1);
                close (cockfd2);
         }
```

Signature of
Teacher incharge

```
        write(1, buf, cnt);
        printf("2\n");
         bzero(buf, BUFSIZE);

    }
    printf("client 1disconnected");
}
int main()
{
        struct sockaddr_in serv;
        int listenfd, acceptfd1, acceptfd2, fval;
        bzero(&serv, sizeof(serv));
        serv.sin_family= AF_INET;
        serv.sin_port = htons(PORT);
        serv.sin_addr.s_addr = htonl(IP);
        listenfd = socket (AF_INET, SOCK_STRE
                                    0);
        if (listenfd <0)
          {    printf("unable to create socket"
                  exit(0);
          }
```

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date ______________ Name ______________

Dept./Lab ______________ Class ______________ Expt./No. ______________

Title ______________

```
if (bind (listenfd, (struct sockaddr *) &serv,
        sizeof(serv)) == -1)
  { printf(" Bind error"); exit(0); }

listen ( listenfd , BACKLOG) ;
while(1)
{
      acceptfd1 = accept ( listenfd , NULL, NULL);
      printf(" Accepted from client 1");
    acceptfd2 = accept( listenfd, NULL, NULL);
      if ( acceptfd1 == -1 ||acceptfd2 == -1)
       {
            printf( "Error accepting");
              exit(0);
       }
       servfunc (acceptfd1 , acceptfd2);
}
return 0; } //end.
```

Signature of
Teacher incharge

Date ....

c

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date ________ Name M. C. SOHAN

Dept./Lab ________ Class ________ Expt./No. 5

Title ________

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

#define BUFSIZE 1024
#define PORT 5000
#define IP "127.0.0.1"

int main()
{
        struct sockaddr_in serv;
        char buf[BUFSIZE];
        int sockfd, cnt;
```

Signature of
Teacher incharge

```
bzero(&serv, sizeof(serv));
serv. sin_family=AF_INET
serv. sin_port = htons(PORT);
if( inet_pton (AF_INET, (vaid*) &serv. sin addr)
                          ==0 )
    {
        printf("Error");
            exit(0);
    }

sockfd = socket( AF_INET, SOCK_STREAM,0)
if ( connect(sockfd, (struct sockaddr*)&serv
                 sizeof(serv)) ==-1)
  {
      printf("unable to connect").
          ~~Error~~ exit(0);
  }

printf("Enter messages:\n");

//cont.
```

```
    while (1)
    {
            if ( cnt = read (1, buf, BUFSIZE))
            {
                    printf (" \n sent %s \n", buf);
                    send ( sockfd, buf, cnt, 0);
            }
            cnt = recv( sockfd , buf , BUFSIZE, 0);
             printf( "recv:");
              write (1, buf, cnt);
              printf( "\n");
               bzero( buf, BUFSIZE);
        }
          close (sockfd);
          return (0);
  } // end.
```