```
                        /*Send the message to server*/
                        send(socketDescriptor,sendBuffer,strlen(sendBuffer)+1,0);
                        printf("\nMessage sent !\n");
        }
        }
        else
        {
                while(1)
                {
                        bzero(&recvBuffer,sizeof(recvBuffer));
                        /*Receive the message from server*/
                        recv(socketDescriptor,recvBuffer,sizeof(recvBuffer),0);
                        printf("\nSERVER : %s\n",recvBuffer);
                }
        }
        return 0;
        }
```

Output:

Server:

```
RA1911026010118:~/environment/RA1911026010114/CN LAB 7/Server (master) $ cc server.c
RA1911026010118:~/environment/RA1911026010114/CN LAB 7/Server (master) $ ./a.out
Server is running ...

Type a message here ...
CLIENT : hello


CLIENT : amuls
```

Client:

```
RA1911026010118:~/environment/RA1911026010114/CN LAB 7/Client (master) $ cc client.c
RA1911026010118:~/environment/RA1911026010114/CN LAB 7/Client (master) $ ./a.out

Type a message here ...  hello

Message sent !

Type a message here ...  amuls

Message sent !

Type a message here ...
```

Result:

Thus the chat application full duplex communication is established by sending the request from the client to the server, server gets the message and gives response to the client and prints it.

Ex.No:8	IMPLEMENTATION OF FILE TRANSFER PROTOCOL
Date:	

Aim :

There are two hosts, Client and Server. The Client sends the name of the file it needs from the Server and the Server sends the contents of the file to the Client, where it is stored in a file.

TECHNICAL OBJECTIVE:

To implement FTP application, where the Client on establishing a connection with the Server sends the name of the file it wishes to access remotely. The Server then sends the contents of the file to the Client, where it is stored.

METHODOLOGY:

Server :

- Include the necessary header files.
- Create a socket using socket function with family AF_INET, type as SOCK_STREAM.
- Initialize server address to 0 using the bzero function.
- Assign the sin_family to AF_INET, sin_addr to INADDR_ANY, sin_port to dynamically assigned port number.
- Bind the local host address to socket using the bind function.
- Listen on the socket for connection request from the client.
- Accept connection request from the Client using accept function.
- Within an infinite loop, receive the file from the client.
- Open the file, read the file contents to a buffer and send the buffer to the Client.

TCP Client :

- Include the necessary header files.
- Create a socket using socket function with family AF_INET, type as SOCK_STREAM.
- Initialize server address to 0 using the bzero function.
- Assign the sin_family to AF_INET, sin_addr to INADDR_ANY, sin_port to dynamically assigned port number.
- Bind the local host address to socket using the bind function.
- Listen on the socket for connection request from the client.
- Accept connection request from the Client using accept function.
- Within an infinite loop, send the name of the file to be viewed to the Server.
- Read the file contents, store it in a file and print it on the console.

Codes :

Server:

```
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
//headers for socket and related functions
#include <sys/types.h>
```

```
#include <sys/socket.h>
#include <sys/stat.h>
//for including structures which will store information needed
#include <netinet/in.h>
#include <unistd.h>
//for gethostbyname
#include "netdb.h"
#include "arpa/inet.h"
// defining constants
#define PORT 6969
#define BACKLOG 5
int main()
{
        int size;
        int socketDescriptor = socket(AF_INET, SOCK_STREAM, 0);
        struct sockaddr_in serverAddress, clientAddress;
        socklen_t clientLength;
        struct stat statVariable;
        char buffer[100], file[1000];
        FILE *filePointer;
        bzero(&serverAddress, sizeof(serverAddress));
        serverAddress.sin_family = AF_INET;
        serverAddress.sin_addr.s_addr = htonl(INADDR_ANY);
       serverAddress.sin_port = htons(PORT);
        bind(socketDescriptor, (struct sockaddr *)&serverAddress, sizeof(serverAddress));
       listen(socketDescriptor,BACKLOG);
       printf("%s\n","Server is running ...");
       int clientDescriptor = accept(socketDescriptor,(struct
       sockaddr*)&clientAddress,&clientLength);
       while(1){
                bzero(buffer,sizeof(buffer));
                bzero(file,sizeof(file));
                recv(clientDescriptor,buffer,sizeof(buffer),0);
                filePointer = fopen(buffer,"r");
                stat(buffer,&statVariable);
                size=statVariable.st_size;
                fread(file,sizeof(file),1,filePointer);
                send(clientDescriptor,file,sizeof(file),0);
}
       return 0;
}
```

Client:

```
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
```

```
//headers for socket and related functions
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/stat.h>
//for including structures which will store information needed
#include <netinet/in.h>
#include <unistd.h>
//for gethostbyname
#include "netdb.h"
#include "arpa/inet.h"
// defining constants
#define PORT 6969
int main()
{
        int serverDescriptor = socket(AF_INET, SOCK_STREAM, 0);
        struct sockaddr_in serverAddress;
        char buffer[100], file[1000];
        bzero(&serverAddress, sizeof(serverAddress));
        erverAddress.sin_family = AF_INET;
        serverAddress.sin_addr.s_addr = inet_addr("127.0.0.1");
        serverAddress.sin_port = htons(PORT);
        connect(serverDescriptor,(struct sockaddr*)&serverAddress,sizeof(serverAddress));
while (1){
        printf("File name : ");
        scanf("%s",buffer);
        send(serverDescriptor,buffer,strlen(buffer)+1,0);
        printf("%s\n","File Output : ");
        recv(serverDescriptor,&file,sizeof(file),0);
        printf("%s",file);
        }
return 0;
}
```

Output:

Server: