EX.NO: 3
DATE:

TRANSMISSION CONTROL PROTOCOL (TCP) SOCKET APPLICATIONS:

A. SIMPLE MESSAGE TRANSFER FROM THE CLIENT TO THE SERVER:

AIM

To send a simple message from the client to the server using the TCP Socket Programming in C.

ALGORITHM:

SERVER PROGRAM:

1. Start.
2. Create sockets named socketmain and socket client.
3. Initialize the socket as with IPv4 (AF_INET) and TCP (SOCK_STREAM)
4. Bind the socket to the sockaddr_in structure.
5. Listen for the client's attempt to initialize the contact.
6. Accept the contact.
7. Read and print the message from the client.
8. Write the message back.
9. Close the sockets.
10. Stop.

CLIENT PROGRAM:

1. Start.
2. Accept the server's host name and the port number as command line arguments.
3. Create sockets named sockfd.
4. Initialize the socket as with IPv4 (AF_INET) and TCP (SOCK_STREAM)
5. Fill the sockaddr_in structures.
6. Connect this socket with the server socket.
7. Once connected, print it.
8. Scan a message.
9. Write it into the server.
10. Get a reply.
11. Print it.
12. Stop.

SOURCE CODE:

SERVER PROGRAM:

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

int main ()
{
        int socketmain, socketclient, port=5000, len=81;
        struct sockaddr_in serv, clientaddr;
        socklen_t clientlen;

        if  ((socketmain = socket(AF_INET, SOCK_STREAM, 0)) < 0)
        {
                printf ("\n Server cannot open the socket");
                exit(0);
        }

        bzero (&serv, sizeof(serv));
        serv.sin_family = AF_INET;
        serv.sin_addr.s_addr = htonl (INADDR_ANY);
        serv.sin_port = htons(port);

        if ( (bind(socketmain,(struct sockaddr*) &serv, sizeof(serv))) < 0)
        {
                printf ("\n Server Bind Failed");
                exit(0);
        }

        listen (socketmain,5);

        if ((socketclient=accept(socketmain,(struct sockaddr*)&clientaddr,&clientlen)) < 0)
        {
                printf ("\n Client is Bad");
                exit(0);
        }


        char buf[len];
        bzero (buf,len);
```

```
        int n = read (socketclient, buf, 100);
        printf ("\n msg is %s", buf);

        close(socketmain);
        close(socketclient);
        return 0;
}
```

CLIENT PROGRAM:

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<stdlib.h>
#include<unistd.h>
#include<arpa/inet.h>
int main (int argc, char **argv)
{
        int sockfd, n, port=5001;
        struct sockaddr_in serv;
        char buf[100], msg[100];

        if (argc != 3)
        {
                printf ("\n Invalid Format");
                exit(0);
        }

        if  ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
        {
                printf ("\n Socket cannot be opened.");
                exit(0);
        }

        bzero (&serv, sizeof(serv));
        serv.sin_family = AF_INET;
        serv.sin_port = htons(atoi(argv[2]));

        if (connect (sockfd, (struct sockaddr*) &serv, sizeof(serv)) < 0)
        {
                printf ("\n Connection Failed.");
                exit(0);
        }

        char i[100];
        printf ("\n Connected");
```

```
        printf ("\n Message");
        scanf("%s",buf);
        write (sockfd,buf,100);
        n = read (sockfd,msg,100);


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

OUTPUT:

SERVER PROGRAM	CLIENT PROGRAM
cc server.c -o s.out ./s.out	cc client.c -o c.out ./c.out localhost 5000
msg is Try	Connected Message Try

RESULT

A simple message from the client to the server using the TCP Socket has been sent successfully.