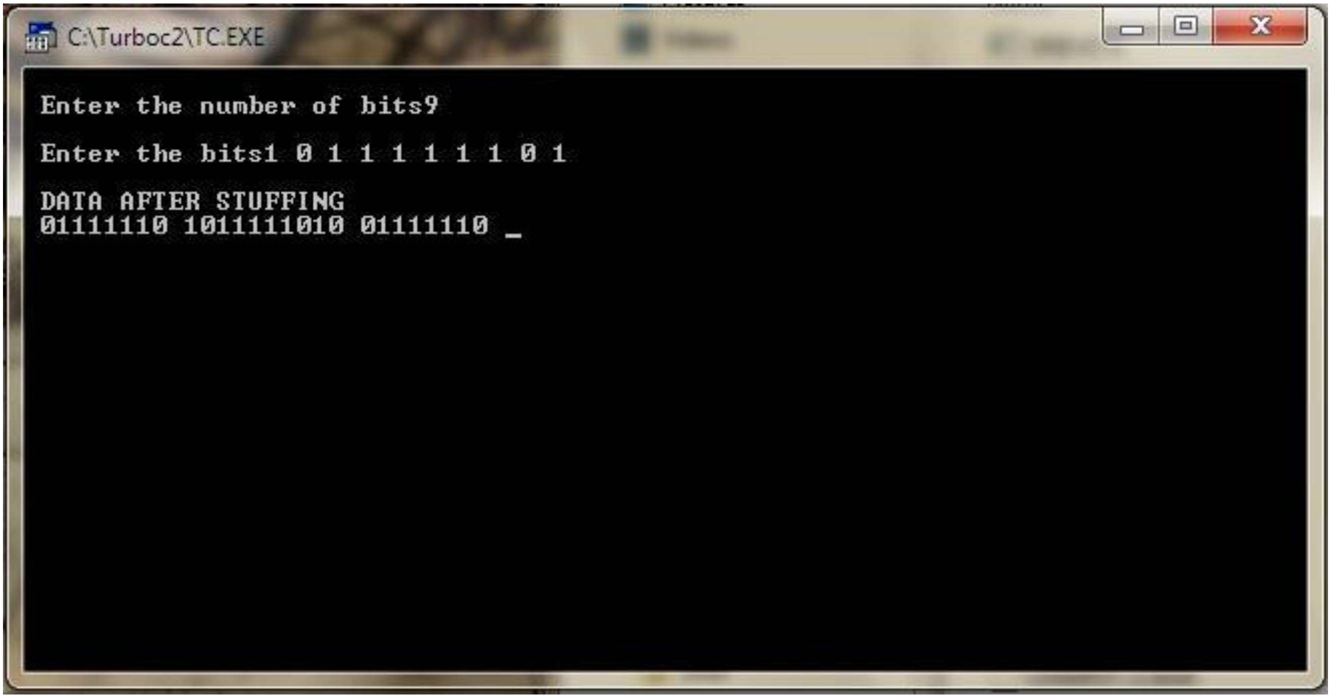
1) Aim:
Implement the data link layer framing methods such as Bit Stuffing.

Theory
Security and Error detection are the most prominent features that are to be provided by any application which transfers data from one end to the other end. One of such a mechanism in tracking errors which may add up to the original data during transfer is known as Stuffing. It is of two types namely Bit Stuffing and the other Character Stuffing. Coming to the Bit Stuffing, 01111110 is appended within the original data while transfer of it. The following program describes how it is stuffed at the sender end and de-stuffed at the reciever end.

Program:

```
#include<string.h>
main()
{
int a[15];
int i,j,k,n,c=0,pos=0;
clrscr();
printf("\n Enter the number of bits");
scanf("%d",&n);
printf("\n Enter the bits");
for(i=0;i<n;i++)
 scanf("%d",&a[i]);
for(i=0;i<n;i++)
 {
if(a[i]==1)
{
c++;
if(c==5)
{
pos=i+1;
c=0;
for(j=n;j>=pos;j--)
{
k=j+1;
a[k]=a[j];
}
a[pos]=0;
n=n+1;
}
}
else
c=0;
}
printf("\n DATA AFTER STUFFING \n");
printf(" 01111110 ");

for(i=0;i<n;i++)
 {
printf("%d",a[i]);
}
printf(" 01111110 ");
getch();
}
```

Output:



```
Enter the number of bits9

Enter the bits1 0 1 1 1 1 1 1 0 1

DATA AFTER STUFFING
01111110 1011111010 01111110 _
```

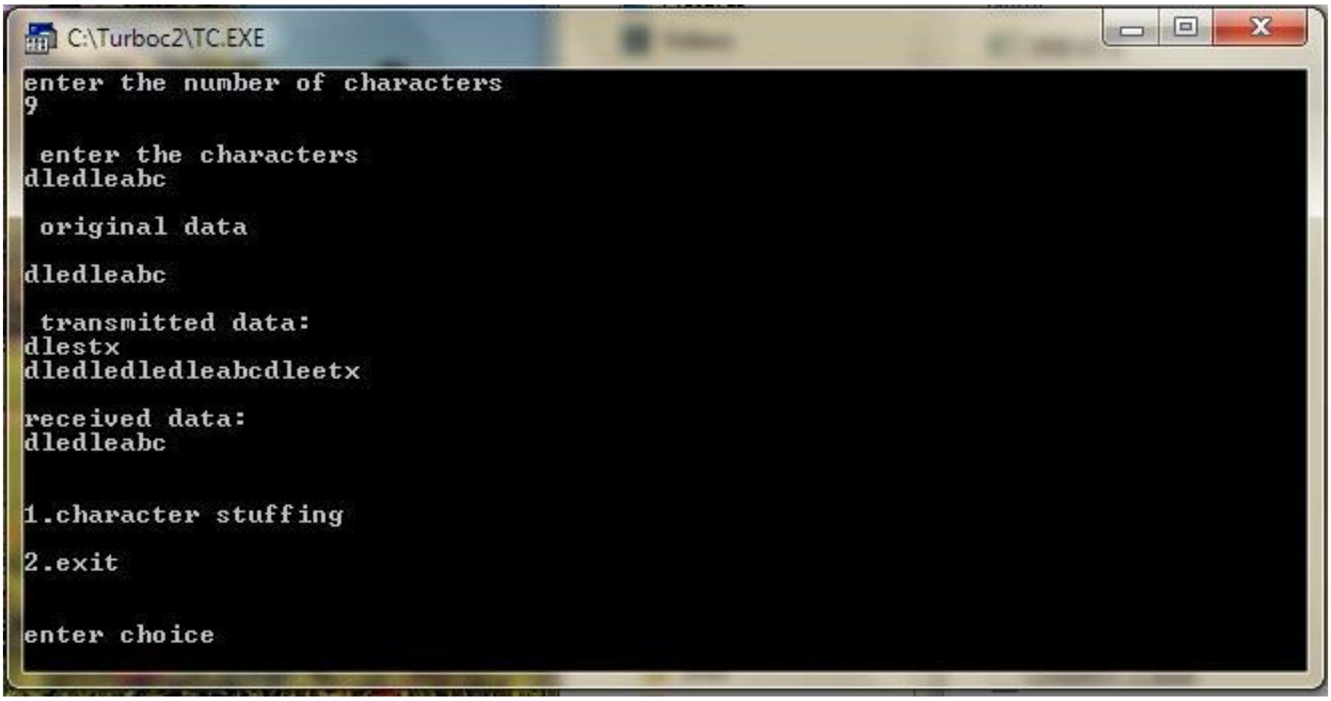
2) Aim:
Implement the data link layer framing methods such as Character Stuffing and also De-stuff it

<u>Theory</u>
Coming to the Character Stuffing, DLESTX and DLEETX are used to denote start and end of character data with some constraints imposed on repetition of charaters as shown in the program below clearly.

Program:

```
#include<stdio.h>
#include<conio.h>

#include<stdlib.h>
void charc(void);
void main()
{
int choice;
while(1)
{
printf("\n\n\n1.character stuffing");
printf("\n\n2.exit");
printf("\n\n\nenter choice");
scanf("%d",&choice);
printf("%d",choice);
if(choice>2)
printf("\n\n invalid option....please renter");
switch(choice)
{
case 1:
charc();
break;
case 2:
exit(0);
}
}
}
void charc(void)
{
char c[50],d[50],t[50];
int i,m,j;
clrscr();
printf("enter the number of characters\n");
scanf("%d",&m);
printf("\n enter the characters\n");
for(i=0;i<m+1;i++)
 {
scanf("%c",&c[i]);
}
printf("\n original data\n");
for(i=0;i<m+1;i++)
 printf("%c",c[i]);
d[0]='d';
d[1]='l';
d[2]='e';
d[3]='s';
d[4]='t';
d[5]='x';
for(i=0,j=6;i<m+1;i++,j++)
 {
if((c[i]=='d'&&c[i+1]=='l'&& c[i+2]=='e'))
{
d[j]='d';
j++;
d[j]='l';
j++;
d[j]='e';
j++;
m=m+3;
}
d[j]=c[i];
}
m=m+6;
m++;
d[m]='d';
m++;
d[m]='l';
m++;
d[m]='e';
m++;
d[m]='e';
m++;
d[m]='t';
m++;
d[m]='x';
m++;
printf("\n\n transmitted data: \n");
for(i=0;i<m;i++)
 {
printf("%c",d[i]);
}
for(i=6,j=0;i<m-6;i++,j++)
 {
if(d[i]=='d'&&d[i+1]=='l'&&d[i+2]=='e'&&d[i+3]=='d'&&d[i+4]=='l'&&d[i+5]=='e')
i=i+3;
t[j]=d[i];
}
printf("\n\nreceived data:");
for(i=0;i<j;i++)
 {printf("%c",t[i]);
}
}
```

Output:



```
enter the number of characters
9

 enter the characters
dledleabc

 original data

dledleabc

 transmitted data:
dlestx
dledledledleabcdleetx

received data:
dledleabc


1.character stuffing

2.exit


enter choice
```

3) Aim:
Implement on a data set of characters the CRC polynomials.

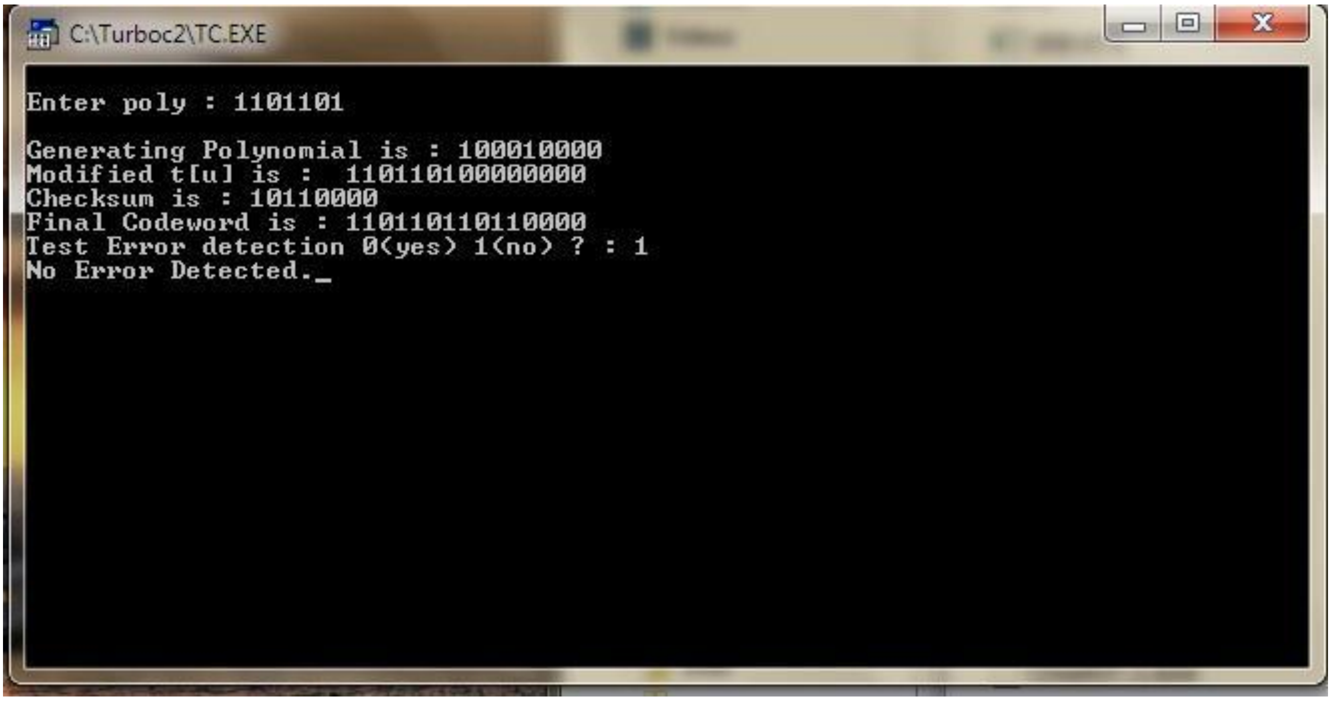
<u>Theory</u>
CRC means Cyclic Redundancy Check. It is the most famous and traditionally successful mechanism used in error detection through the parity bits installed within the data and obtaining checksum which acts as the verifier to check whether the data retreived at the reciever end is genuine or not. Various operations are involved in implementing CRC on a data set through CRC generating polynomials. In the program, I have also provided the user to opt for Error detection whereby he can proceed for it. Understand the program below as it is much simpler than pretented to be so.

Program:

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#define N strlen(g)

char t[128], cs[128], g[]="100010000";
int a, e, c;

void xor() {
for(c=1;c}

void crc() {
for(e=0;e do {
if(cs[0]=='1') xor();
```

```
for(c=0;c cs[c]=t[e++];
}while(e<=a+N-1);
}

void main() {
clrscr();
printf("\nEnter poly : "); scanf("%s",t);
printf("\nGenerating Polynomial is : %s",g);
a=strlen(t);
for(e=a;e
 printf("\nModified t[u] is : %s",t);
crc();
printf("\nChecksum is : %s",cs);
for(e=a;e printf("\nFinal Codeword is : %s",t);
printf("\nTest Error detection 0(yes) 1(no) ? : ");
scanf("%d",&e);
if(e==0) {
printf("Enter position where error is to inserted : ");
scanf("%d",&e);
t[e]=(t[e]=='0')?'1':'0';
printf("Errorneous data : %s\n",t);
}
crc();
for (e=0;(e<n-1)&&(cs[e]!='1');e++);
 if(e
 else printf("No Error Detected.");
getch();
}
```

Output:



```
Enter poly : 1101101

Generating Polynomial is : 100010000
Modified t[u] is :  110110100000000
Checksum is : 10110000
Final Codeword is : 110110110110000
Test Error detection 0(yes) 1(no) ? : 1
No Error Detected._
```

4) Aim:
Implement Dijkstra's algorithm to compute the Shortest path through a graph.

Theory
Dijkstra's algorithm is a non-adaptive routing algorithm which is very widely used to route packets from source to detination through various routers available during the transmission. It is implemented at the network layer of the architecture where data packets are sent through routers which maitain routing tables that help to denote the exact location to where the destined packets need to be delivered. Major advantage in using Dijkstra's algorithm is that it forwards the data packets from source to destination through the most optimized path in terms of both the distance and cost observed. It prompts the user to enter the number of nodes and the source and destination nodes among them. In addition, the algorithm written below also asks for the neighbours to each node with the distances to reach to them from each node is also prompted. All this data is stored and used further to calculate and estimate the best path possible for data packets to reach their destination from source. Program below explains it in a much better way.

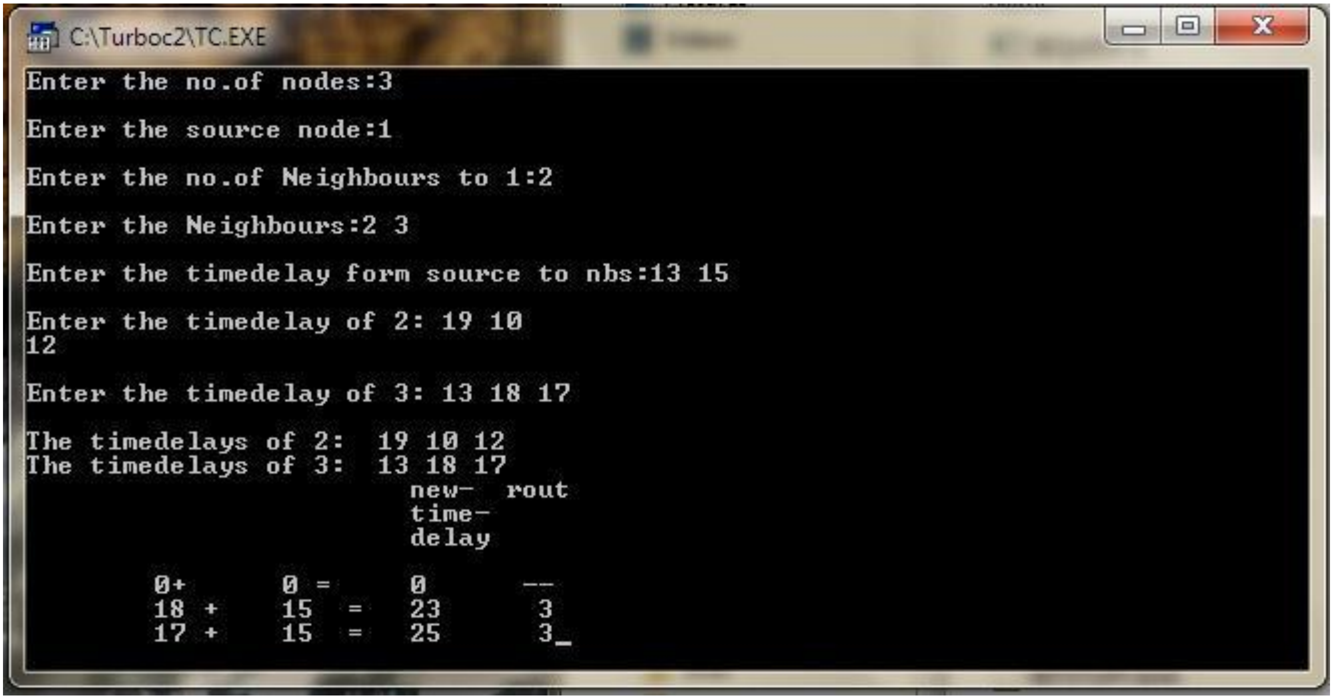
Program:

```
#include<stdlib.h>
#include<conio.h>
int n,s,nb,nbs[15],snbs[15],delay[15][15],i,j,temp[15],ze=0;
void min();
void main()
{
clrscr();
printf("Enter the no.of nodes:");
scanf("%d",&n);
```

```
printf("\nEnter the source node:");
scanf("%d",&s);
printf("\nEnter the no.of Neighbours to %d:",s);
scanf("%d",&nb);
printf("\nEnter the Neighbours:");
for(i=1;i<=nb;i++)
scanf("%d",&nbs[i]);
printf("\nEnter the timedelay form source to nbs:");
for(i=1;i<=nb;i++)
scanf("%d",&snbs[i]);
for(i=1;i<=nb;i++)
{
printf("\nEnter the timedelay of %d: ",nbs[i]);
for(j=1;j<=n;j++)
scanf("%d",&delay[i][j]);
}
for(i=1;i<=nb;i++)
{
printf("\nThe timedelays of %d: ",nbs[i]);
for(j=1;j<=n;j++)
printf("%3d",delay[i][j]);
}
min();
getch();
}
void min()
{
int sum,k,y=1,store=1;
printf("\n\t\t\tnew- rout");
printf("\n\t\t\ttime-");
printf("\n\t\t\tdelay");
printf("\n");
for(i=1;i<=n;i++)
{
sum=0;
k=1;
for(j=1;j<=nb;j++)
{
temp[k++]=delay[j][i];
}

sum=temp[1]+snbs[1];
for(y=2;y<=nb;y++)
{
if(sum>temp[y]+snbs[y])
{
```

```
sum=temp[y]+snbs[y];
store=y;
}
}

if(s==i)
printf("\n\t%d+\t%d =\t%d --",ze,ze,ze);
else
printf("\n\t%d +\t%d =\t%d\t%d",temp[store],snbs[store],sum,nbs[store]);
}
}
```

Output:



```
Enter the no.of nodes:3

Enter the source node:1

Enter the no.of Neighbours to 1:2

Enter the Neighbours:2 3

Enter the timedelay form source to nbs:13 15

Enter the timedelay of 2: 19 10
12

Enter the timedelay of 3: 13 18 17

The timedelays of 2:  19 10 12
The timedelays of 3:  13 18 17
                        new-  rout
                        time-
                        delay

        0+      0 =     0      --
        18 +    15  =   23      3
        17 +    15  =   25      3
```

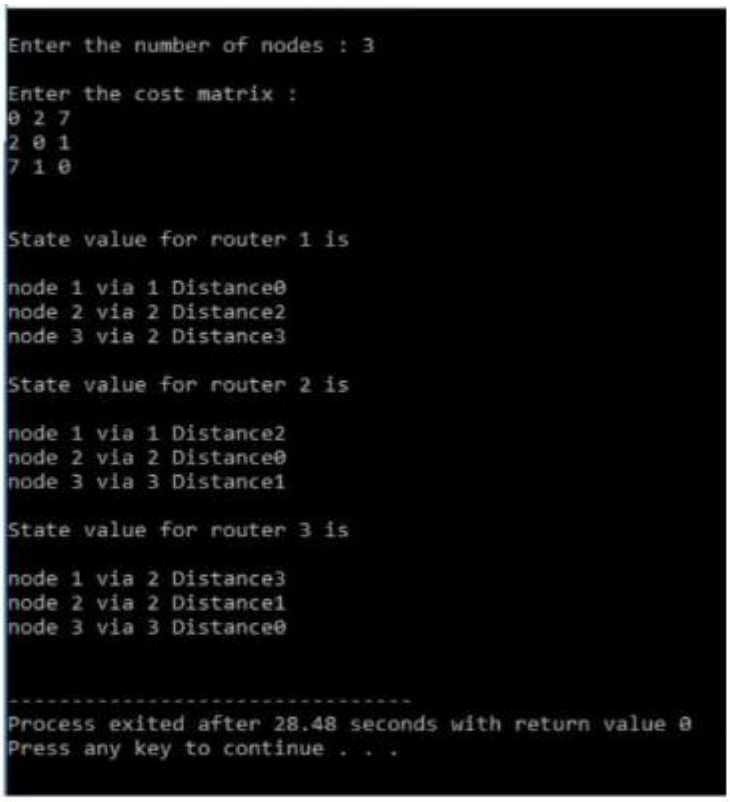
5) Aim:
Take an example subnet graph with weights indicating delay between nodes. Now obtain Routing table at each node using Distance Vector Routing.

<u>Theory</u>

Distance Vector routing (DVR) algorithm is unlike Dijkstra's algorithm which is a non-adaptive routing algorithm and means that it is purely static, that is pre-destined and fixed, not flexible in networks where congestions are more prone to occur. DVR is an adaptive routing algorithm in which the information from neighbours is maitained well by each and every node and this helps us to determine the simplest path possible in a changing network. Though, one of the node may fail, still, the destined node is reachable through other possible intermediate nodes that are found out by the DVR algorithm. The perfectly executing program below shows it live below.

Program:

```
#include<stdlib.h>

struct node
{
unsigned dist[20];
unsigned from[20];
}rt[10];

int main()
{
int dmat[20][20];
int n,i,j,k,count=0;
printf("\nEnter the number of nodes : ");
scanf("%d",&n);
printf("\nEnter the cost matrix :\n");
for(i=0;i<n;i++)
 for(j=0;j<n;j++)
 {
scanf("%d",&dmat[i][j]);
dmat[i][i]=0;
rt[i].dist[j]=dmat[i][j];
rt[i].from[j]=j;
}
do
{
count=0;
for(i=0;i<n;i++)
 for(j=0;j<n;j++)
 for(k=0;k<n;k++)
 if(rt[i].dist[j]>dmat[i][k]+rt[k].dist[j])
{
rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];
rt[i].from[j]=k;
count++;
}
}while(count!=0);
for(i=0;i<n;i++)
 {
printf("\n\nState value for router %d is \n",i+1);
for(j=0;j<n;j++)
 {
printf("\t\nnode %d via %d Distance%d",j+1,rt[i].from[j]+1,rt[i].dist[j]);
}
}
printf("\n\n");
}
```



```
Enter the number of nodes : 3

Enter the cost matrix :
0 2 7
2 0 1
7 1 0


State value for router 1 is

node 1 via 1 Distance0
node 2 via 2 Distance2
node 3 via 2 Distance3

State value for router 2 is

node 1 via 1 Distance2
node 2 via 2 Distance0
node 3 via 3 Distance1

State value for router 3 is

node 1 via 2 Distance3
node 2 via 2 Distance1
node 3 via 3 Distance0



--------------------------------
Process exited after 28.48 seconds with return value 0
Press any key to continue . . .
```

6) Aim:
Take an example subnet of hosts. Obtain broadcast tree for it.

Theory
IP addressing is the allocation of unique ID to each and every system connected in a network to maintan communication among them through out the affixed network. There are 5 classes of IP Addresses namely A through E with the range varying from one class to the other class. A subnet is a network allocation to similar systems or same hierarchial systems present in a allocated network like an organisation. Each and every system can be reachd through a client-server computing environment where the server acts as the Master and the clients acts as the Slaves to form a Master-Slave computing environment. Below programs show the calculation of network addresses with subnet predefinition and subnet generation.

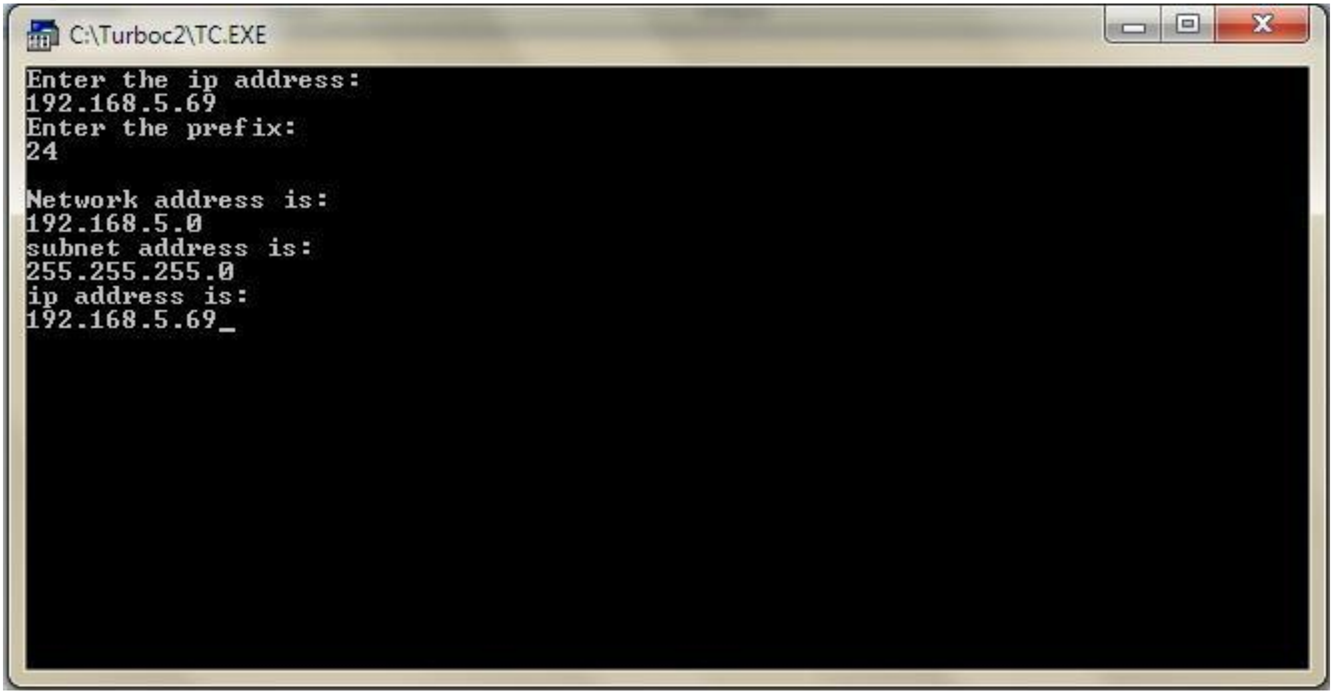
Program:

a)Network Address:

```
#include<conio.h>
#include<stdlib.h>
void main()
```

```
{
unsigned int compad[4];
unsigned int mask[4];
unsigned int netadr[4];
int i;
clrscr();
printf("Enter the ip address:\n");
scanf("%u%*c%u%*c%u%*c%u%*c",&compad[3],&compad[2],&compad[1],&compad[0])
;
printf("Enter the subnet address:\n");
scanf("%u%*c%u%*c%u%*c%u%*c",&mask[3],&mask[2],&mask[1],&mask[0]);
for(i=0;i<4;i++)
{
netadr[i]= compad[i]&mask[i];
}
printf("\nNetwork address is:\n");
printf("%u.%u.%u.%u",netadr[3],netadr[2],netadr[1],netadr[0]);
printf("\nsubnet address is:\n");
printf("%u.%u.%u.%u",mask[3],mask[2],mask[1],mask[0]);
printf("\nip address is:\n");
printf("%u.%u.%u.%u",compad[3],compad[2],compad[1],compad[0]);
getch();
}
```

Output:



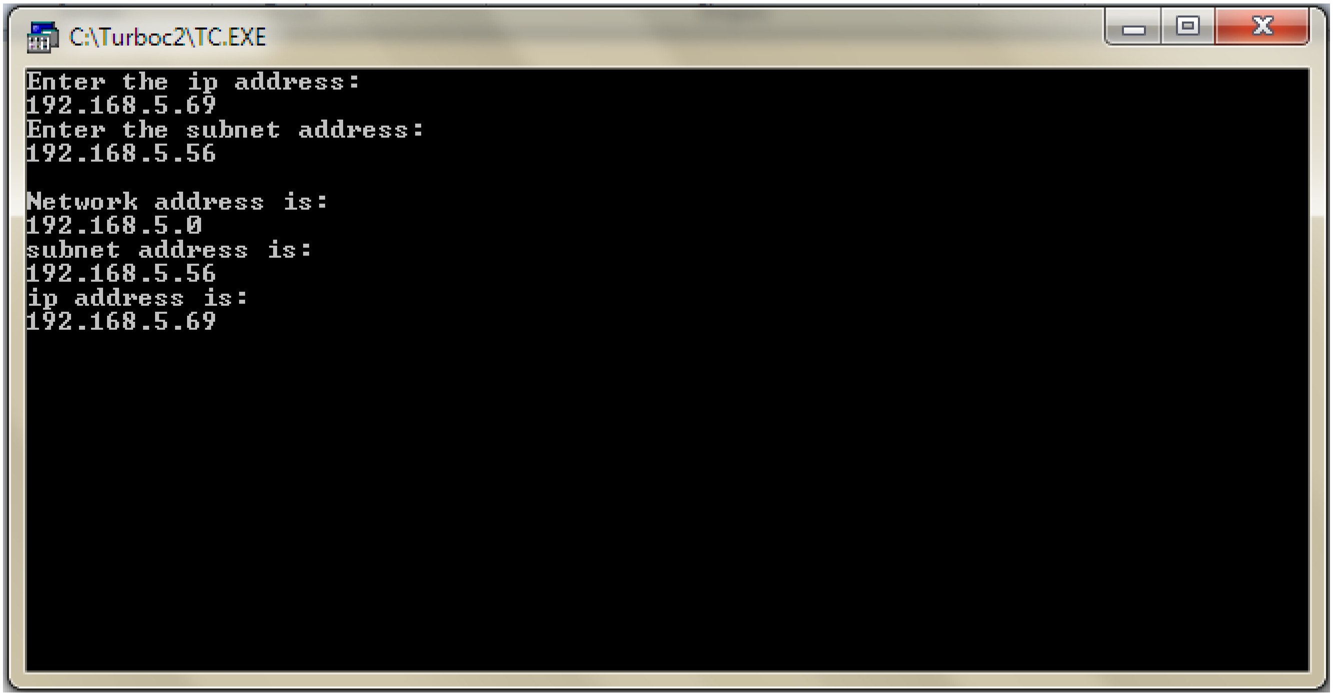
```
Enter the ip address:
192.168.5.69
Enter the prefix:
24

Network address is:
192.168.5.0
subnet address is:
255.255.255.0
ip address is:
192.168.5.69_
```

b)Network address with automatic subnet address generation:

```
#include<conio.h>
#include<stdlib.h>
```

```
void main()
{
unsigned int compad[4];
unsigned int mask[4];
unsigned int netadr[4];
unsigned long int ma=0;
int i,pre;
clrscr();
printf("Enter the ip address:\n");
scanf("%u%*c%u%*c%u%*c%u%*c",&compad[3],&compad[2],&compad[1],&compad[0])
;
printf("Enter the prefix:\n");
scanf("%ul",&pre);
for(i=(32-pre);i<32;i++)
ma=ma|(1<<i);
 for(i=0;i<4;i++)
{
mask[i]=ma%256;
ma=ma/256;
}
for(i=0;i<4;i++)
{
netadr[i]= compad[i]&mask[i];
}
printf("\nNetwork address is:\n");
printf("%u.%u.%u.%u",netadr[3],netadr[2],netadr[1],netadr[0]);
printf("\nsubnet address is:\n");
printf("%u.%u.%u.%u",mask[3],mask[2],mask[1],mask[0]);
printf("\nip address is:\n");
printf("%u.%u.%u.%u",compad[3],compad[2],compad[1],compad[0]);
getch();
}
```

Output:



```
Enter the ip address:
192.168.5.69
Enter the subnet address:
192.168.5.56

Network address is:
192.168.5.0
subnet address is:
192.168.5.56
ip address is:
192.168.5.69
```

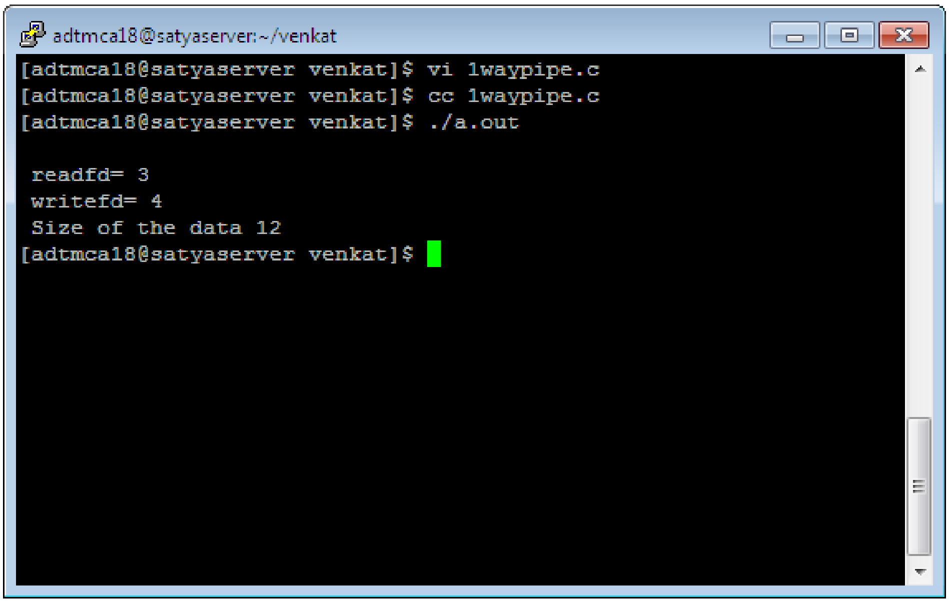
1.Implement the following forms of IPC.

a) Pipes

One way communication in one process

```
#include<stdio.h>
#include<stdlib.h>
main()
{
 int pipefd[2],n;
 char buff[100];
 pipe(pipefd);
  printf("\n readfd= %d", pipefd[0]);
  printf("\n writefd= %d", pipefd[1]);
  write(pipefd[1],"Hello World",12);
  n=read(pipefd[0],buff, sizeof(buff));
 printf("\n Size of the data %d",n);
 printf("\n data from pipe: %s",buff);
}
```

OUTPUT :



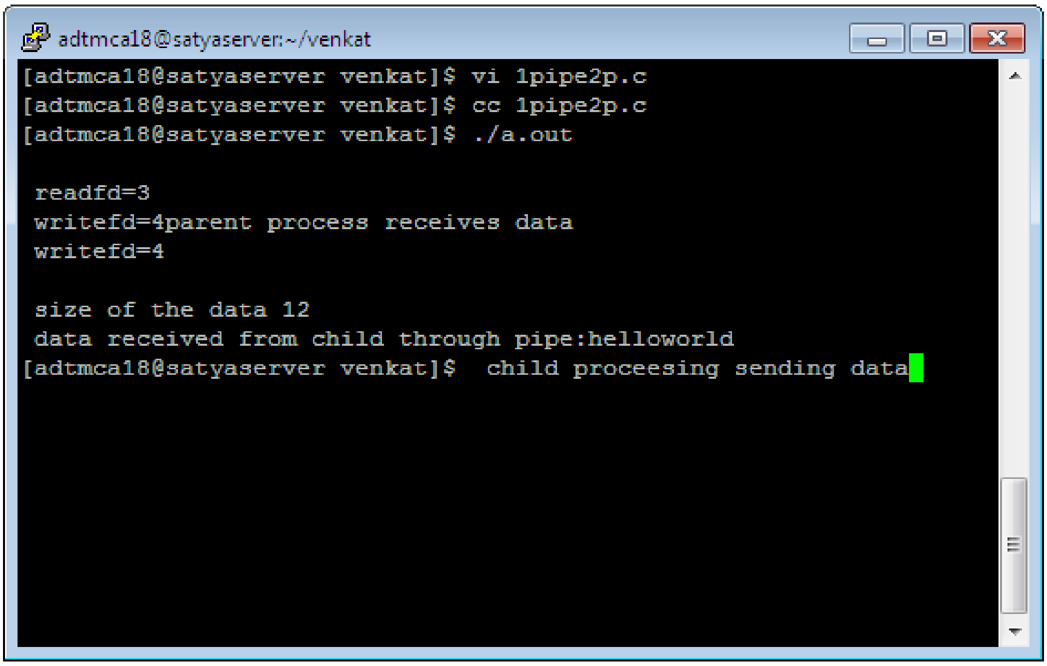
```
[adtmca18@satyaserver venkat]$ vi 1waypipe.c
[adtmca18@satyaserver venkat]$ cc 1waypipe.c
[adtmca18@satyaserver venkat]$ ./a.out

 readfd= 3
 writefd= 4
 Size of the data 12
[adtmca18@satyaserver venkat]$
```

One way communication in between two process

```
#include<stdio.h>
#include<stdlib.h>
```

```
main()
{
int pipefd[2],n,pid;
char buff[100];
pipe(pipefd);
printf("\n readfd=%d",pipefd[0]);
printf("\n writefd=%d",pipefd[1]);
pid=fork();
if(pid==0)
{
close(pipefd[0]);
printf("\n child proceesing sending data");
write(pipefd[1],"helloworld",12);
}
else
{
close(pipefd[1]);
printf("parent process receives data\n");
n=read(pipefd[0],buff,sizeof(buff));
printf("\n size of the data %d",n);
printf("\n data received from child through pipe:%s \n",buff);
}
}
```

OUTPUT :

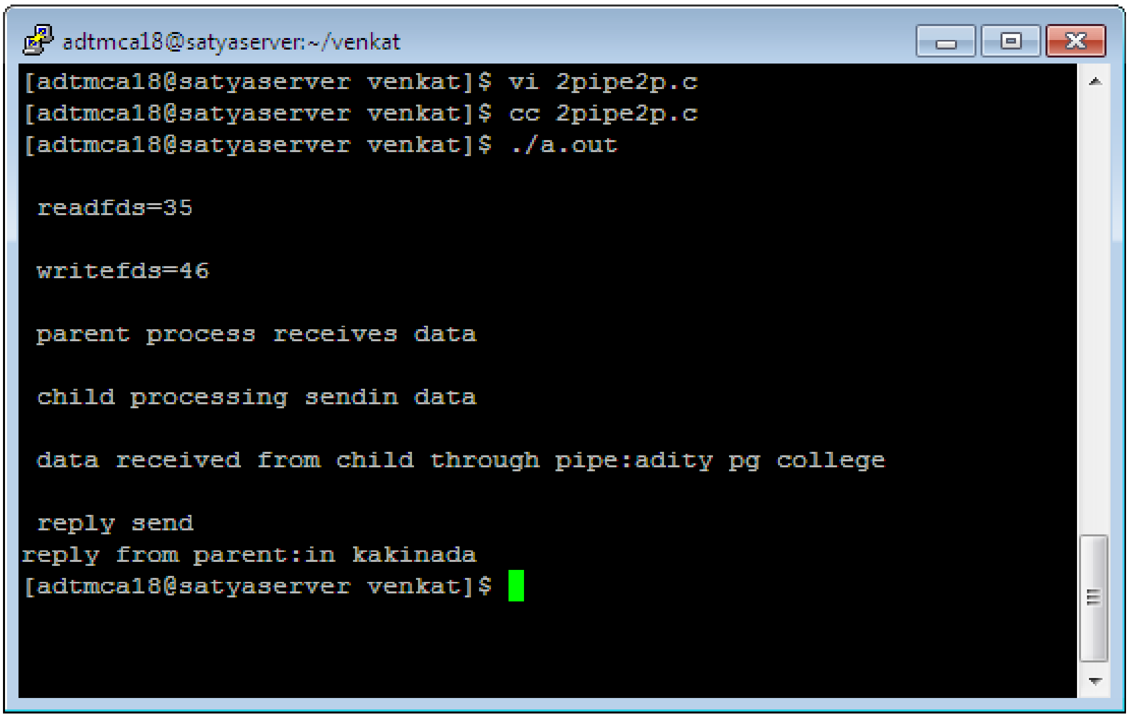


```
[adtmca18@satyaserver venkat]$ vi 1pipe2p.c
[adtmca18@satyaserver venkat]$ cc 1pipe2p.c
[adtmca18@satyaserver venkat]$ ./a.out

 readfd=3
 writefd=4parent process receives data
 writefd=4

 size of the data 12
 data received from child through pipe:helloworld
[adtmca18@satyaserver venkat]$  child proceesing sending data
```

Two way communication in between two process

```
#include<stdio.h>
#include<stdlib.h>
main()
{
  int p1[2],p2[2],n,pid;
  char buff1[25],buff2[25];
  pipe(p1);
  pipe(p2);
 printf("\n readfds=%d%d \n",p1[0],p2[0]);
 printf("\n writefds=%d%d \n",p1[1],p2[1]);
 pid=fork();
 if(pid==0)
 {
  close(p1[0]);
  printf("\n child processing sendin data\n");
  write(p1[1],"adity pg college",25);
  close(p2[1]);
  read(p2[0],buff1,25);
  printf("reply from parent:%s \n",buff1);
  sleep(2);
 } else
{
   close(p1[1]);
   printf("\n parent process receives data \n");
   n=read(p1[0],buff2,sizeof(buff2));
   printf("\n data received from child through pipe:%s \n",buff2);
   sleep(3);
   close(p2[0]);
   write(p2[1],"in kakinada",25);
  printf("\n reply send \n");
  } }
```

OUTPUT :



```
[adtmca18@satyaserver venkat]$ vi 2pipe2p.c
[adtmca18@satyaserver venkat]$ cc 2pipe2p.c
[adtmca18@satyaserver venkat]$ ./a.out

 readfds=35

 writefds=46

 parent process receives data

 child processing sendin data

 data received from child through pipe:adity pg college

 reply send
reply from parent:in kakinada
[adtmca18@satyaserver venkat]$ 
```

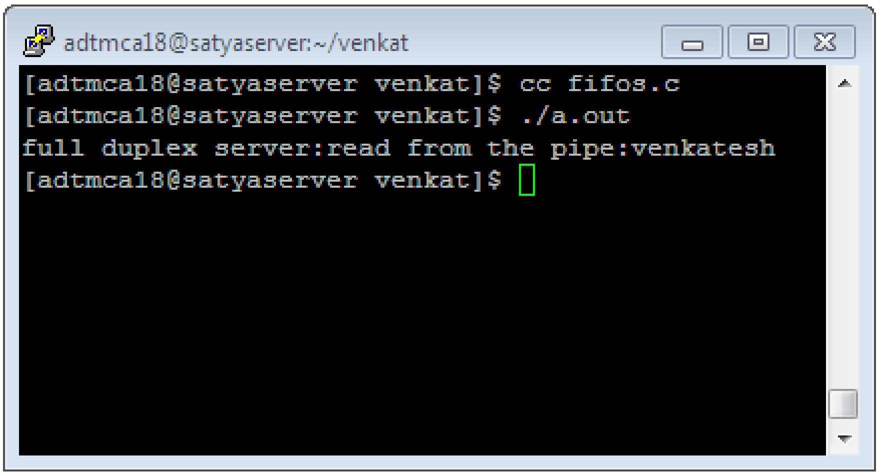
b) FIFO

Write a program to demonstrate inter process communication through fifo between client and server

Server program

```
#include<stdio.h>
#include<ctype.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<stdlib.h>
#include<string.h>
main()
{
int wrfd,rdfd,n,d,ret_val,count;
char buf[50];
ret_val=mkfifo("np1",0666);
ret_val=mkfifo("np2",0666);
rdfd=open("np1",O_RDONLY);
wrfd=open("np2",O_WRONLY);
n=read(rdfd,buf,50);
buf[n]='\0';
printf("full duplex server:read from the pipe:%s\n",buf);
count=0;
while(count<n)
{
buf[count]=toupper(buf[count]);
count++;
}
write(wrfd,buf,strlen(buf));
}
```

OUTPUT :

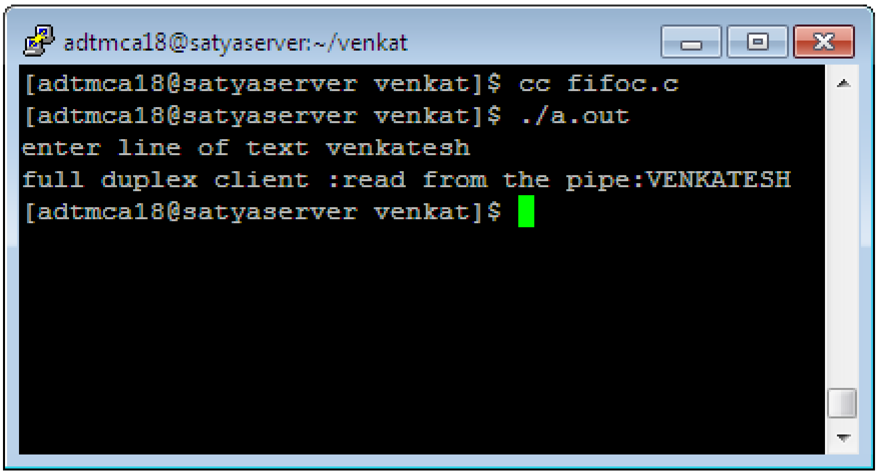


```
[adtmca18@satyaserver venkat]$ cc fifos.c
[adtmca18@satyaserver venkat]$ ./a.out
full duplex server:read from the pipe:venkatesh
[adtmca18@satyaserver venkat]$
```

Client Program

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>
main()
{
int wrfd,rdfd,n;
char buf[50],line[50];
wrfd=open("np1",O_WRONLY);
rdfd=open("np2",O_RDONLY);
printf("enter line of text");
write(wrfd,line,strlen(line));
n=read(rdfd,buf,50);
buf[n]='\0';
printf("full duplex client:read from the pipe:%s\n",buf);
}
```

OUTPUT :



```
[adtmca18@satyaserver venkat]$ cc fifoc.c
[adtmca18@satyaserver venkat]$ ./a.out
enter line of text venkatesh
full duplex client :read from the pipe:VENKATESH
[adtmca18@satyaserver venkat]$
```

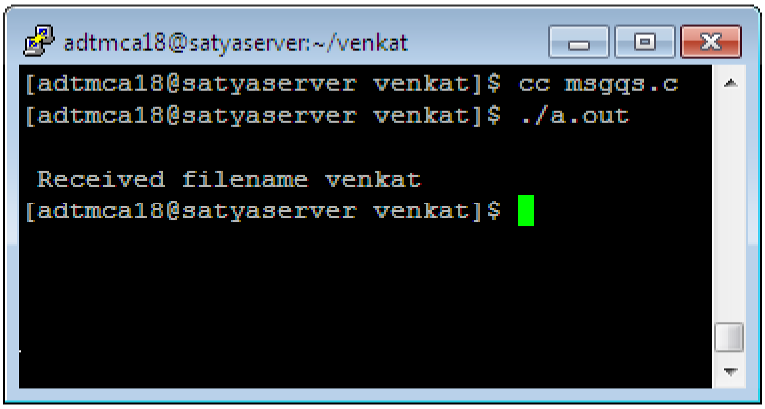
2.Implement file transfer using Message Queue form of IPC.

Server Program

```
#include<stdio.h>
#include<ctype.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<stdlib.h>
#include<string.h>
main()
{
int wrfd,rdfd,n,d,ret_val,count;
char buf[50];
ret_val=mkfifo("np1",0666);
ret_val=mkfifo("np2",0666);
rdfd=open("np1",O_RDONLY);
wrfd=open("np2",O_WRONLY);
n=read(rdfd,buf,50);
buf[n]='\0';
printf("full duplex server:read from the pipe:%s\n",buf);
count=0;
while(count<n)
{
```

```
buf[count]=toupper(buf[count]);
count++;
}
write(wrfd,buf,strlen(buf));
}
```

OUTPUT :

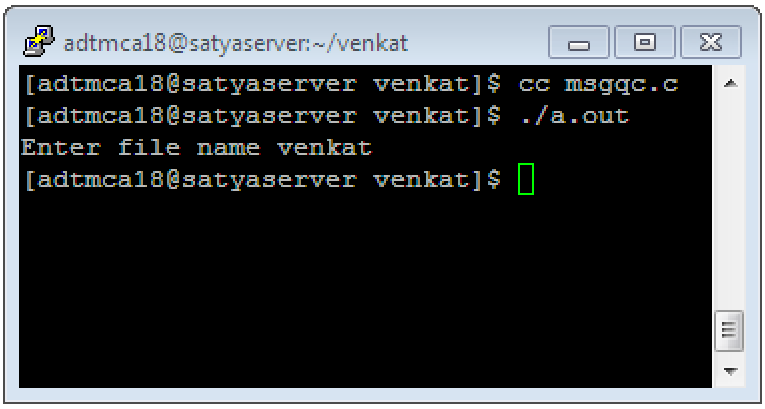


```
[adtmca18@satyaserver venkat]$ cc msgqs.c
[adtmca18@satyaserver venkat]$ ./a.out

 Received filename venkat
[adtmca18@satyaserver venkat]$
```

Client Program

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>
main()
{
int wrfd,rdfd,n;
char buf[50],line[50];
wrfd=open("np1",O_WRONLY);
rdfd=open("np2",O_RDONLY);
printf("enter line of text");
write(wrfd,line,strlen(line));
n=read(rdfd,buf,50);
buf[n]='\0';
printf("full duplex client:read from the pipe:%s\n",buf);
}
```

OUTPUT :



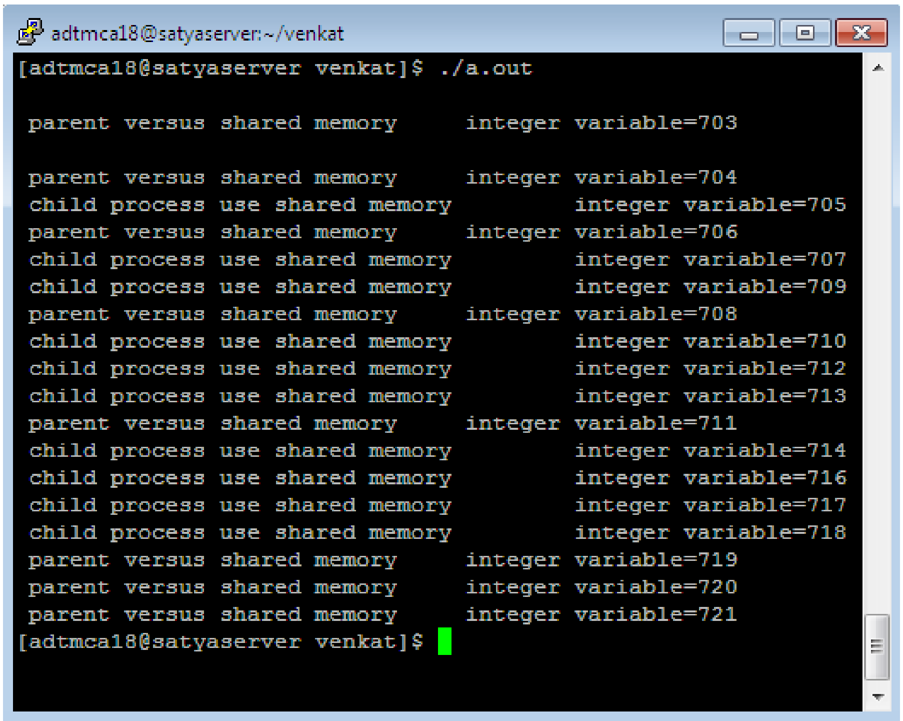
```
[adtmca18@satyaserver venkat]$ cc msgqc.c
[adtmca18@satyaserver venkat]$ ./a.out
Enter file name venkat
[adtmca18@satyaserver venkat]$
```

3.Write a Program to create an integer variable using Shared Memory concept and increment the variable simultaneously by two processes. Use Semaphores to avoid Race conditions.

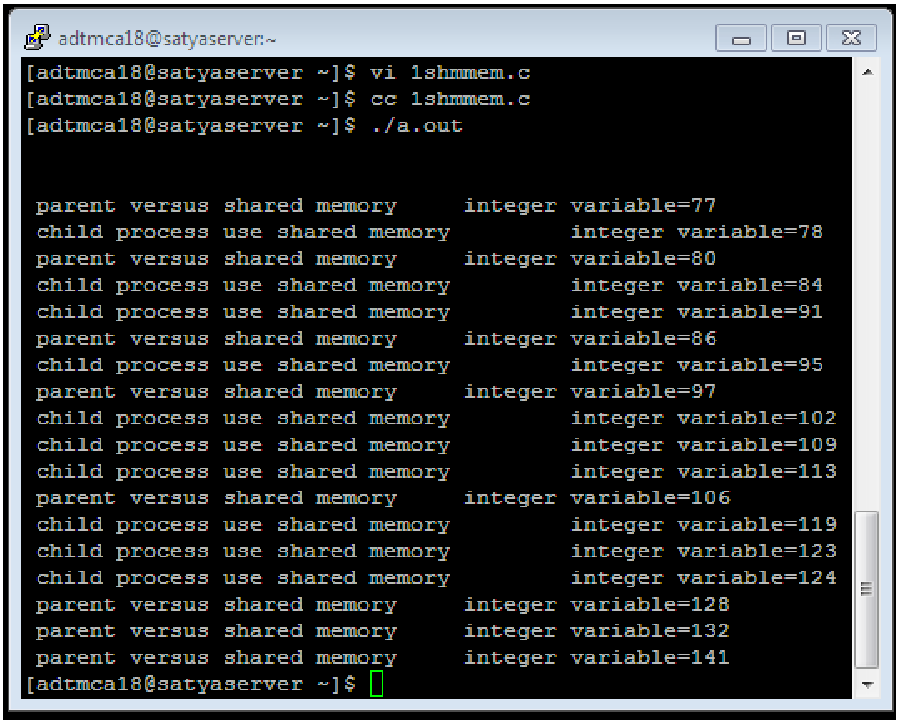
```
#include<sys/stat.h>
#include<stdio.h>
#include<sys/types.h>
#include<sys/shm.h>
#include<sys/ipc.h>
#include<sys/sem.h>
#include<string.h>
#define SIZE 10
int *integer=0;
main()
{
int shmid;
key_t key_10;
char *shm;
int semid,pid;
shmid=shmget((key_t)10,SIZE,IPC_CREAT|0666);
shm=shmat(shmid,NULL,0);
semid = semget(0X20,1,IPC_CREAT|0666);
integer=(int *)shm;
```

```
pid=fork();
if(pid==0)
{
int i=0;
while(i<10)
{
sleep(2);
printf("\n child process use shared memory");
accessmem(semid);
i++;
}
}
else
{
int j=0;
while(j<10)
{
sleep(j);
printf("\n parent versus shared memory");
accessmem(semid);
j++;
}
}
shmctl(semid,IPC_RMID,0);
}
int accessmem(int semid)
{
struct sembuf sop;
sop.sem_num=0;
sop.sem_op=-1;
sop.sem_flg=0;
semop(semid,&sop,1);
(*integer)++;
printf("\t integer variable=%d",(*integer));
sop.sem_num=0;
sop.sem_op=1;
sop.sem_flg=0;
semop(semid,&sop,1);
}
```

OUTPUT :



```
[adtmca18@satyaserver venkat]$ ./a.out

 parent versus shared memory     integer variable=703

 parent versus shared memory     integer variable=704
 child process use shared memory         integer variable=705
 parent versus shared memory     integer variable=706
 child process use shared memory         integer variable=707
 child process use shared memory         integer variable=709
 parent versus shared memory     integer variable=708
 child process use shared memory         integer variable=710
 child process use shared memory         integer variable=712
 child process use shared memory         integer variable=713
 parent versus shared memory     integer variable=711
 child process use shared memory         integer variable=714
 child process use shared memory         integer variable=716
 child process use shared memory         integer variable=717
 child process use shared memory         integer variable=718
 parent versus shared memory     integer variable=719
 parent versus shared memory     integer variable=720
 parent versus shared memory     integer variable=721
[adtmca18@satyaserver venkat]$ 
```



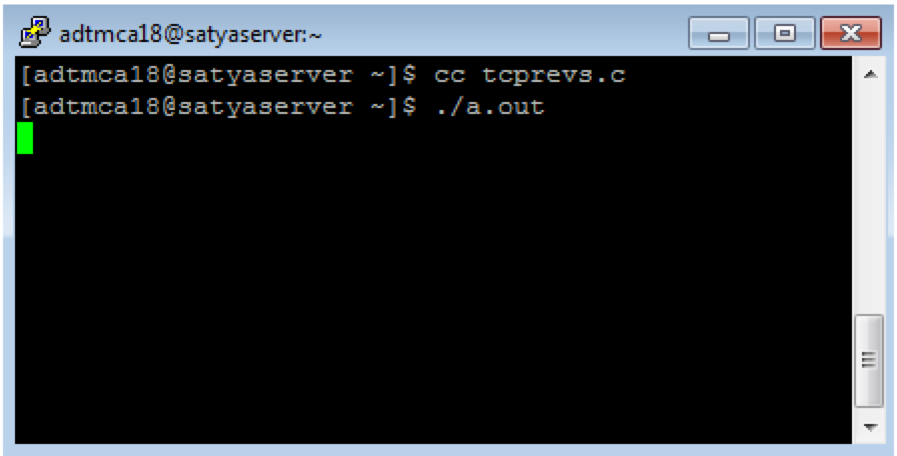
```
[adtmca18@satyaserver ~]$ vi 1shmmem.c
[adtmca18@satyaserver ~]$ cc 1shmmem.c
[adtmca18@satyaserver ~]$ ./a.out


 parent versus shared memory     integer variable=77
 child process use shared memory         integer variable=78
 parent versus shared memory     integer variable=80
 child process use shared memory         integer variable=84
 child process use shared memory         integer variable=91
 parent versus shared memory     integer variable=86
 child process use shared memory         integer variable=95
 parent versus shared memory     integer variable=97
 child process use shared memory         integer variable=102
 child process use shared memory         integer variable=109
 child process use shared memory         integer variable=113
 parent versus shared memory     integer variable=106
 child process use shared memory         integer variable=119
 child process use shared memory         integer variable=123
 child process use shared memory         integer variable=124
 parent versus shared memory     integer variable=128
 parent versus shared memory     integer variable=132
 parent versus shared memory     integer variable=141
[adtmca18@satyaserver ~]$ 
```

4. Design TCP iterative Client and Server application to reverse the given input sentence.

Server Program

```
#include<string.h>
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<sys/types.h>
#define MAXLINE 20
#define SERV_PORT 5777
main(int argc,char *argv)
{
int i,j;
ssize_t n;
char line[MAXLINE],revline[MAXLINE];
int listenfd,connfd,clilen;
struct sockaddr_in servaddr,cliaddr;
listenfd=socket(AF_INET,SOCK_STREAM,0);
bzero(&servaddr,sizeof(servaddr));
servaddr.sin_family=AF_INET;
servaddr.sin_port=htons(SERV_PORT);
bind(listenfd,(struct sockaddr*)&servaddr,sizeof(servaddr));
listen(listenfd,1);
for( ; ; )
{
clilen=sizeof(cliaddr);
connfd=accept(listenfd,(struct sockaddr*)&cliaddr,&clilen);
printf("connect to client");
while(1)
{
if((n=read(connfd,line,MAXLINE))==0)
break;

line[n-1]='\0';
j=0;
for(i=n-2;i>=0;i--)
revline[j++]=line[i];
revline[j]='\0';
write(connfd,revline,n);
}
}
}
```

OUTPUT :



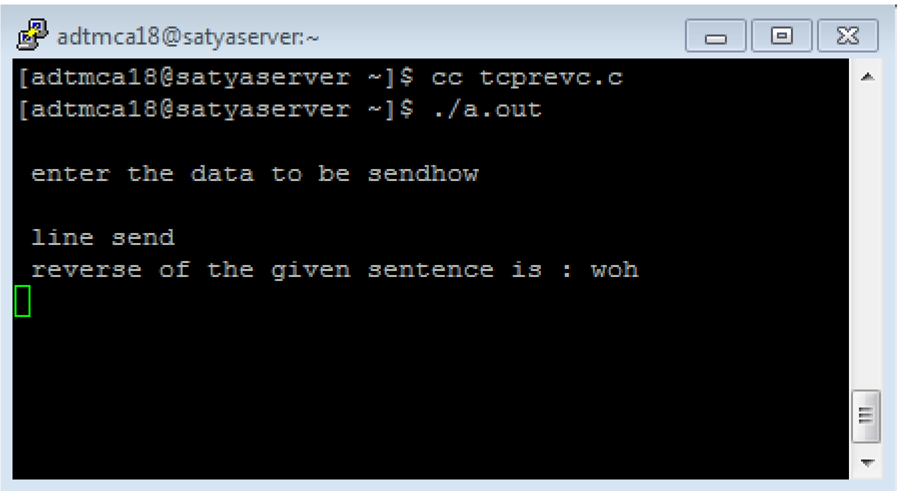
```
[adtmca18@satyaserver ~]$ cc tcprevs.c
[adtmca18@satyaserver ~]$ ./a.out
```

Client Program

```
#include<string.h>
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<sys/types.h>
#define MAXLINE 20
#define SERV_PORT 5777
main(int argc,char *argv)
{
```

```
char sendline[MAXLINE],revline[MAXLINE];
int sockfd;
struct sockaddr_in servaddr;
sockfd=socket(AF_INET,SOCK_STREAM,0);
bzero(&servaddr,sizeof(servaddr));
servaddr.sin_family=AF_INET;
servaddr.sin_port=ntohs(SERV_PORT);
connect(sockfd,(struct sockaddr*)&servaddr,sizeof(servaddr));
printf("\n enter the data to be send");
while(fgets(sendline,MAXLINE,stdin)!=NULL)
{
write(sockfd,sendline,strlen(sendline));
printf("\n line send");
read(sockfd,revline,MAXLINE);
printf("\n reverse of the given sentence is : %s",revline);
printf("\n");
}
exit(0);
}
```

OUTPUT :



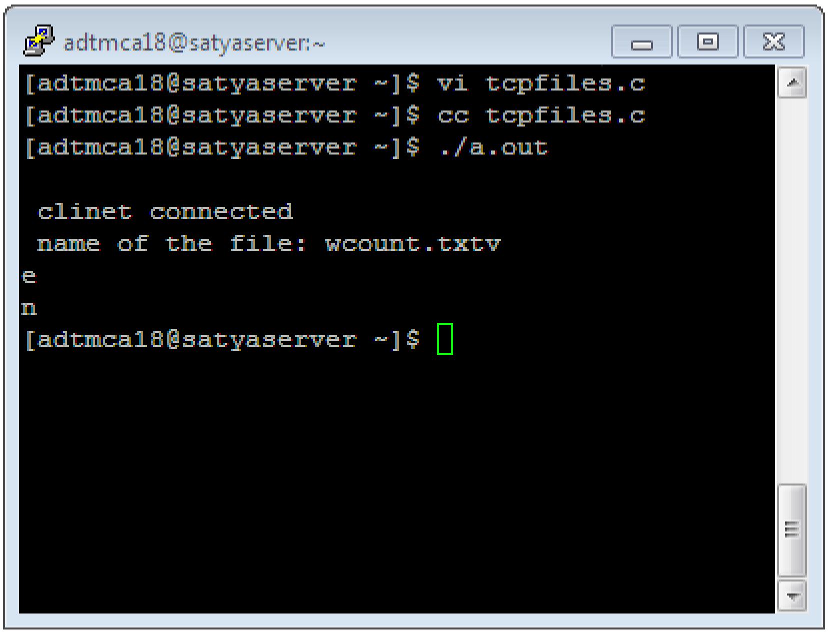
```
[adtmca18@satyaserver ~]$ cc tcprevc.c
[adtmca18@satyaserver ~]$ ./a.out

 enter the data to be sendhow

 line send
 reverse of the given sentence is : woh
```

5.Design TCP client and server application to transfer file

Server program:

```
#include<stdio.h>
#include<unistd.h>
#include<string.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<sys/types.h>
#define SERV_PORT 5576
main(int argc,char **argv)
{
int i,j;  ssize_t n;
FILE *fp; char s[80],f[80];
struct sockaddr_in servaddr,cliaddr;
int listenfd,connfd,clilen;
listenfd=socket(AF_INET,SOCK_STREAM,0);
bzero(&servaddr,sizeof(servaddr));
servaddr.sin_family=AF_INET;
servaddr.sin_port=htons(SERV_PORT);
bind(listenfd,(struct sockaddr *)&servaddr,sizeof(servaddr));
listen(listenfd,1);
clilen=sizeof(cliaddr);
connfd=accept(listenfd,(struct sockaddr*)&cliaddr,&clilen);
printf("\n clinet connected");
read(connfd,f,80);
fp=fopen(f,"r");
printf("\n name of the file: %s",f);
while(fgets(s,80,fp)!=NULL)
{ printf("%s",s);
write(connfd,s,sizeof(s));
} }
```

OUTPUT :

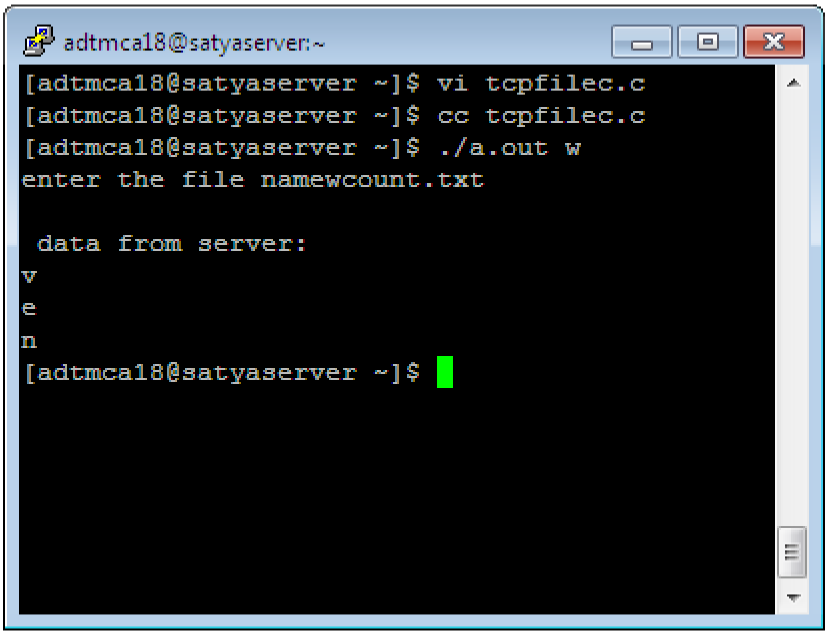


```
[adtmca18@satyaserver ~]$ vi tcpfiles.c
[adtmca18@satyaserver ~]$ cc tcpfiles.c
[adtmca18@satyaserver ~]$ ./a.out

 clinet connected
 name of the file: wcount.txtv
e
n
[adtmca18@satyaserver ~]$ 
```

Client Program :

```
#include<stdio.h>
#include<unistd.h>
#include<string.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<sys/types.h>
#define SERV_PORT 5576
main(int argc,char **argv)
{
int i,j;
ssize_t n;
char filename[80],recvline[80];
struct sockaddr_in servaddr;
int sockfd;
sockfd=socket(AF_INET,SOCK_STREAM,0);
bzero(&servaddr,sizeof(servaddr));
servaddr.sin_family=AF_INET;
servaddr.sin_port=htons(SERV_PORT);
inet_pton(AF_INET,argv[1],&servaddr.sin_addr);
connect(sockfd,(struct sockaddr*)&servaddr,sizeof(servaddr));
printf("enter the file name");
scanf("%s",filename);
write(sockfd,filename,sizeof(filename));
printf("\n data from server: \n");
while(read(sockfd,recvline,80)!=0)
{
fputs(recvline,stdout);
}
}
```

OUTPUT :



```
[adtmca18@satyaserver ~]$ vi tcpfilec.c
[adtmca18@satyaserver ~]$ cc tcpfilec.c
[adtmca18@satyaserver ~]$ ./a.out w
enter the file namewcount.txt

 data from server:
v
e
n
[adtmca18@satyaserver ~]$
```

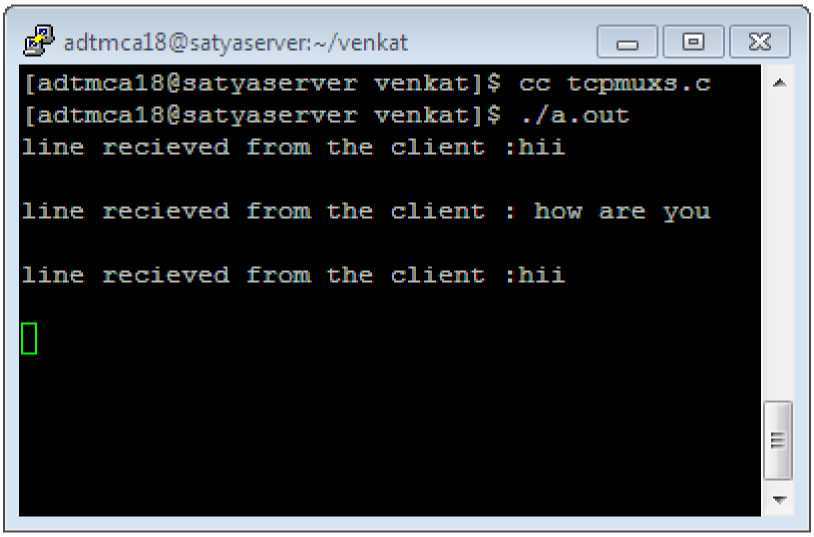
6. Design a TCP concurrent server to convert a given text into upper case using multiplexing system call “select”.

Server Program

```
#include<stdio.h>
#include<netinet/in.h>
#include<sys/types.h>
#include<string.h>
#include<stdlib.h>
#include<sys/socket.h>
#include<sys/select.h>
```

```
#include<unistd.h>
#define MAXLINE 20
#define SERV_PORT 7134
main(int argc,char **argv)
{
int i,j,maxi,maxfd,listenfd,connfd,sockfd;
int nread,client[FD_SETSIZE];
ssize_t n;
fd_set rset,allset;
char line[MAXLINE];
socklen_t clilen;
struct sockaddr_in cliaddr,servaddr;
listenfd=socket(AF_INET,SOCK_STREAM,0);
bzero(&servaddr,sizeof(servaddr));
servaddr.sin_family=AF_INET;
servaddr.sin_port=htons(SERV_PORT);
bind(listenfd,(struct sockaddr *)&servaddr,sizeof(servaddr));
listen(listenfd,1);
maxfd=listenfd;
maxi=-1;
for(i=0;i<FD_SETSIZE;i++)
client[i]=-1;
FD_ZERO(&allset);
FD_SET(listenfd,&allset);
for(; ;)
{
rset=allset;
nread=select(maxfd+1,&rset,NULL,NULL,NULL);
if(FD_ISSET(listenfd,&rset))
{
clilen=sizeof(cliaddr);
connfd=accept(listenfd,(struct sockaddr*)&cliaddr,&clilen);
for(i=0;i<FD_SETSIZE;i++)
if(client[i]<0)
{
 client[i]=connfd; break;
 }
 if(i==FD_SETSIZE)
{
 printf("too many clients");
exit(0);
 }
FD_SET(connfd,&allset);
if(connfd>maxfd) maxfd=connfd;
if(i>maxi)
maxi=i;
if(--nread<=0) continue;
}
for(i=0;i<=maxi;i++)
{
if((sockfd=client[i])<0) continue;
if(FD_ISSET(sockfd,&rset))
{
if((n=read(sockfd,line,MAXLINE))==0)
{
close(sockfd);
FD_CLR(sockfd,&allset);
client[i]=-1;
}
else
{
printf("line recieved from the client :%s\n",line);
for(j=0;line[j]!='\0';j++) line[j]=toupper(line[j]);
write(sockfd,line,MAXLINE);
}
 if(--nread<=0) break;
}
 }
 }
 }
```

OUTPUT :

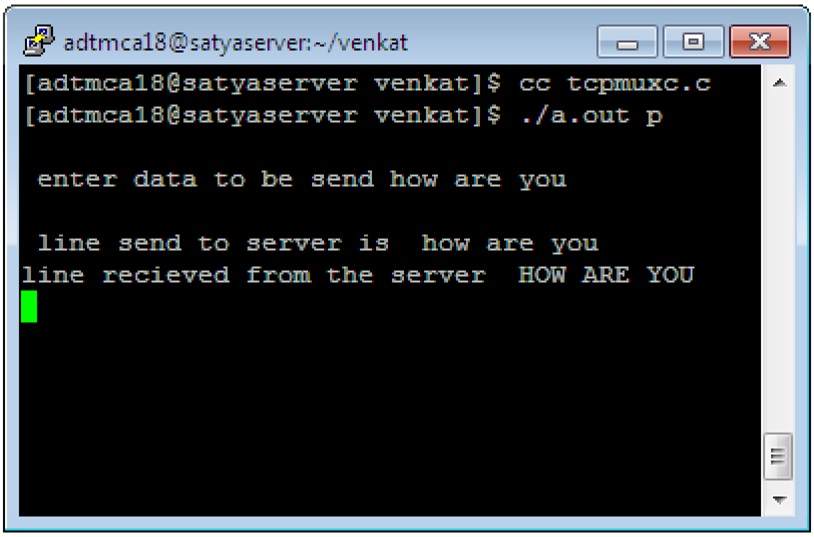


```
[adtmca18@satyaserver venkat]$ cc tcpmuxs.c
[adtmca18@satyaserver venkat]$ ./a.out
line recieved from the client :hii

line recieved from the client : how are you

line recieved from the client :hii

```

Client Program :

```
#include<netinet/in.h>
#include<sys/types.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<sys/socket.h>
#include<sys/select.h>
#include<unistd.h>
#define MAXLINE 20
#define SERV_PORT 7134
main(int argc,char **argv)
{
int maxfdp1;
fd_set rset;
char sendline[MAXLINE],recvline[MAXLINE];
int sockfd;
struct sockaddr_in servaddr;
if(argc!=2)
{
printf("usage tcpcli <ipaddress>");
return;
 }
 sockfd=socket(AF_INET,SOCK_STREAM,0);
 bzero(&servaddr,sizeof(servaddr));
 servaddr.sin_family=AF_INET;
 servaddr.sin_port=htons(SERV_PORT);
 inet_pton(AF_INET,argv[1],&servaddr.sin_addr);
 connect(sockfd,(struct sockaddr*)&servaddr,sizeof(servaddr));
 printf("\n enter data to be send");
 while(fgets(sendline,MAXLINE,stdin)!=NULL)
 {
write(sockfd,sendline,MAXLINE);
 printf("\n line send to server is %s",sendline);
 read(sockfd,recvline,MAXLINE);
 printf("line recieved from the server %s",recvline);
 }
 exit(0);
 }
```

OUTPUT :



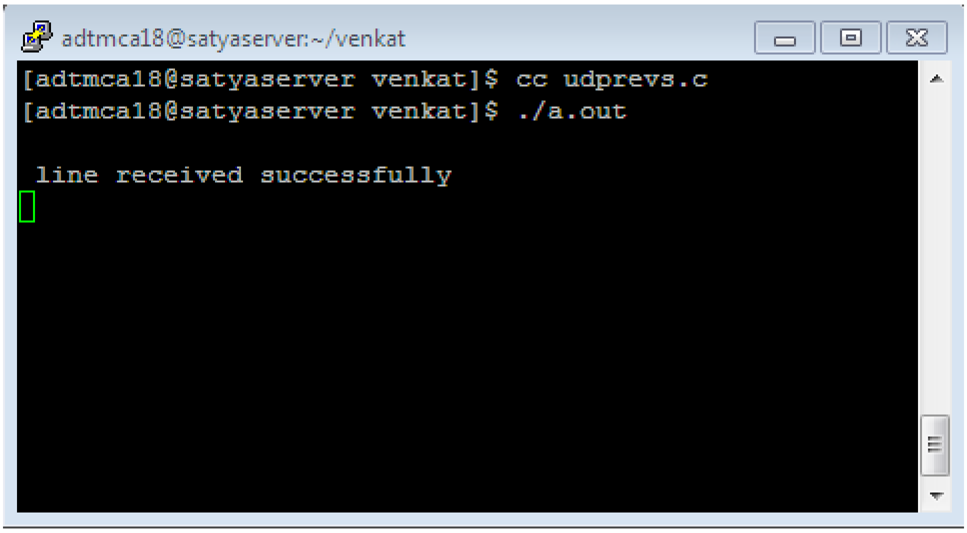
```
[adtmca18@satyaserver venkat]$ cc tcpmuxc.c
[adtmca18@satyaserver venkat]$ ./a.out p

 enter data to be send how are you

 line send to server is  how are you
line recieved from the server  HOW ARE YOU
```

7. Design UDP Client and server application to reverse the given input sentence

Server Program

```
#include<stdio.h>
#include<unistd.h>
#include<string.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<sys/types.h>
#include<stdlib.h>
#define SERV_PORT 5839
#define MAXLINE 20 main(int argc,char **argv)
{
int i,j; ssize_t n;
char line[MAXLINE],recvline[MAXLINE];
struct sockaddr_in servaddr,cliaddr;
int sockfd,clilen;
sockfd=socket(AF_INET,SOCK_DGRAM,0); bzero(&servaddr,sizeof(servaddr));
servaddr.sin_family=AF_INET; servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
servaddr.sin_port=htons(SERV_PORT);
bind(sockfd,(struct sockaddr*)&servaddr,sizeof(servaddr));
for( ; ; )
{
clilen=sizeof(cliaddr);
while(1)
{
if((n=recvfrom(sockfd,line,MAXLINE,0,(struct sockaddr*)&cliaddr,&clilen))==0)
break;
printf("\n line received successfully");
line[n-1]='\0';
j=0; for(i=n-2;i>=0;i--){
recvline[j++]=line[i];
}
recvline[j]='\0';
sendto(sockfd,recvline,n,0,(struct sockaddr*)&cliaddr,clilen);
}
}
}
```

OUTPUT :

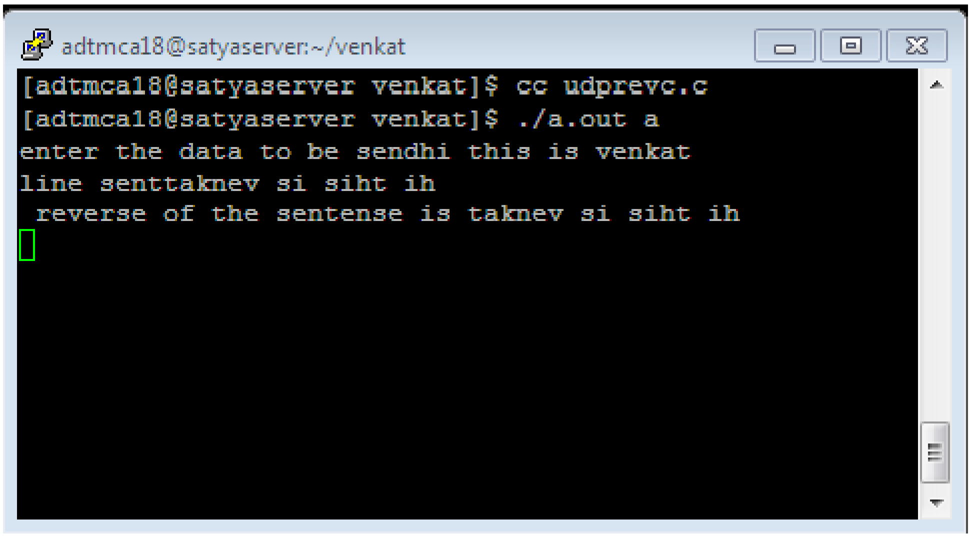


```
[adtmca18@satyaserver venkat]$ cc udprevs.c
[adtmca18@satyaserver venkat]$ ./a.out

 line received successfully
```

Client Program

```
#include<stdio.h>
#include<unistd.h>
#include<string.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<sys/types.h>
#include<stdlib.h>
#define SERV_PORT 5839
#define MAXLINE 20 main(int argc,char **argv)
{
ssize_t n;
struct sockaddr_in servaddr;
char sendline[MAXLINE],recvline[MAXLINE];
int sockfd;
if(argc!=2)
{
printf("usage:<IPADDRESS>");
exit(0);
}
bzero(&servaddr,sizeof(servaddr));
servaddr.sin_family=AF_INET;
servaddr.sin_port=htons(SERV_PORT);
inet_pton(AF_INET,argv[1],&servaddr.sin_addr);
sockfd=socket(AF_INET,SOCK_DGRAM,0);
printf("enter the data to be send");
while(fgets(sendline,MAXLINE,stdin)!=NULL)
{
sendto(sockfd,sendline,strlen(sendline),0,(struct
sockaddr*)&servaddr,sizeof(servaddr)); printf("line sent");
n=recvfrom(sockfd,recvline,MAXLINE,0,NULL,NULL);
recvline[n]='\0';
fputs(recvline,stdout);
printf("\n reverse of the sentense is %s",recvline); printf("\n");
}
exit(0);
}
```

OUTPUT :



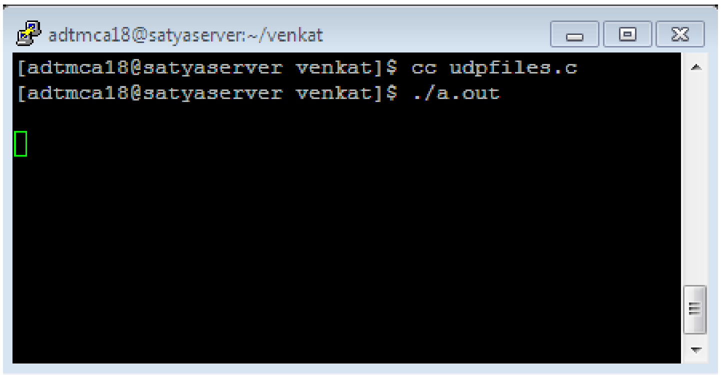
```
[adtmca18@satyaserver venkat]$ cc udprevc.c
[adtmca18@satyaserver venkat]$ ./a.out a
enter the data to be sendhi this is venkat
line senttaknev si siht ih
 reverse of the sentense is taknev si siht ih
```

8. Design UDP Client Server to transfer a file.

Server Program

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#define SERV_PORT 6349 main(int argc,char **argv)
{
char filename[80],recvline[80]; FILE *fp;
struct sockaddr_in servaddr,cliaddr;
int clilen,sockfd;
sockfd=socket(AF_INET,SOCK_DGRAM,0);
bzero(&servaddr,sizeof(servaddr));
servaddr.sin_family=AF_INET;
servaddr.sin_port=htons(SERV_PORT);
bind(sockfd,(struct sockaddr*)&servaddr,sizeof(servaddr));
clilen=sizeof(cliaddr);
recvfrom(sockfd,filename,80,0,(struct sockaddr*)&cliaddr,&clilen);
printf("\n date in the file is \n ");
fp=fopen(filename,"r");
while(fgets(recvline,80,fp)!=NULL)
{
printf("\n %s\n ",recvline);
}fclose(fp);
}
```

OUTPUT :

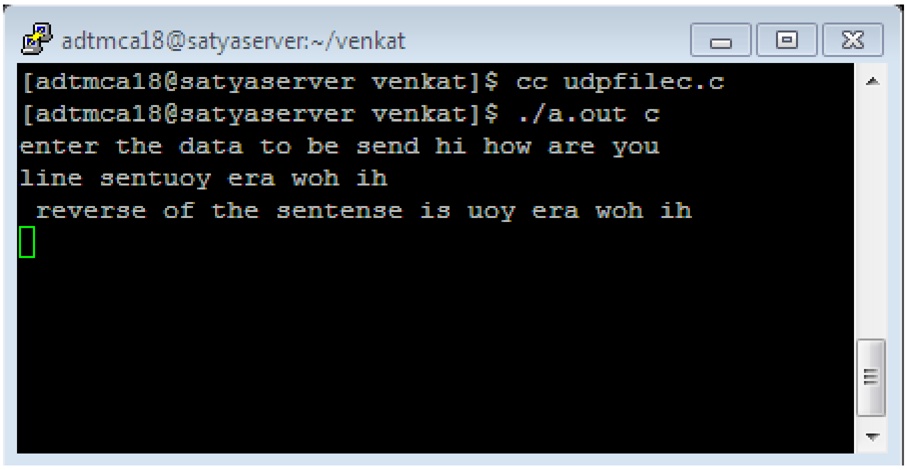


```
[adtmca18@satyaserver venkat]$ cc udpfiles.c
[adtmca18@satyaserver venkat]$ ./a.out
```

Client Program

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<unistd.h>
#define SERV_PORT 6349 main(int argc,char **argv)
{
char filename[80]; int sockfd;
struct sockaddr_in servaddr;
sockfd=socket(AF_INET,SOCK_DGRAM,0);
bzero(&servaddr,sizeof(servaddr));
servaddr.sin_family=AF_INET;
servaddr.sin_port=htons(SERV_PORT);
inet_pton(AF_INET,argv[1],&servaddr.sin_addr);
printf("enter the file name");
scanf("%s",filename);
sendto(sockfd,filename,strlen(filename),0,(structsockaddr*)&servaddr,sizeof
(servaddr))
}
```

OUTPUT :



```
[adtmca18@satyaserver venkat]$ cc udpfilec.c
[adtmca18@satyaserver venkat]$ ./a.out c
enter the data to be send hi how are you
line sentuoy era woh ih
 reverse of the sentense is uoy era woh ih
```

9.Design using poll client server application to multiplex TCP and UDP requests for converting a given text into upper case.

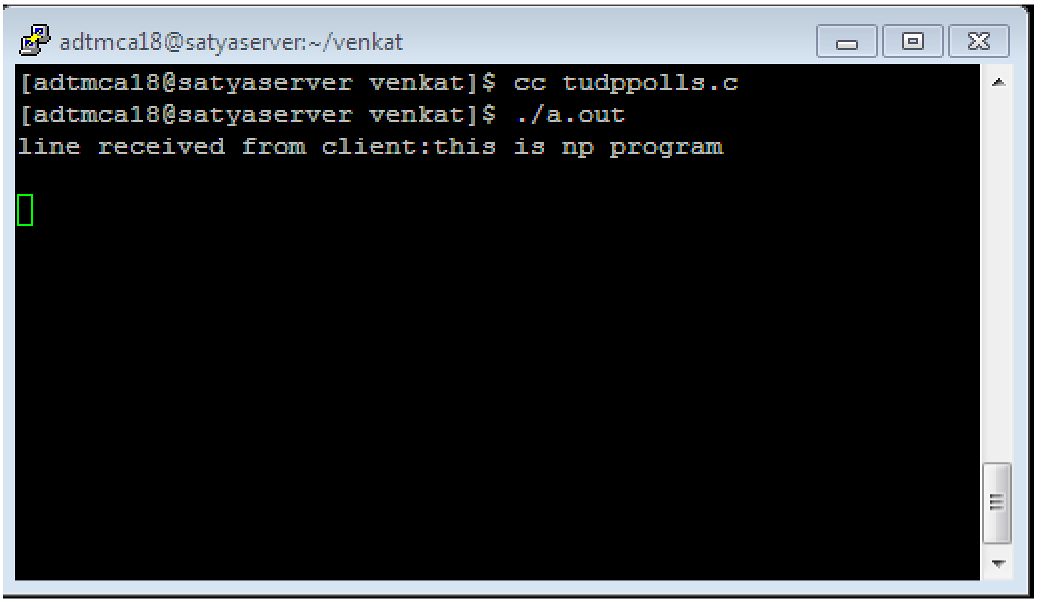
Server Program

```
#include<stdio.h>
#include<netinet/in.h>
#include<sys/types.h>
#include<string.h>
#include<stdlib.h>
#include<sys/socket.h>
#include<sys/select.h>
#include<unistd.h>
#define MAXLINE 20
#define SERV_PORT 8114
main(int argc,char **argv)
{
int i,j,maxi,maxfd,listenfd,connfd,sockfd;
int nready,client[FD_SETSIZE];
ssize_t n;
fd_set rset,allset;
char line[MAXLINE];
socklen_t clilen;
struct sockaddr_in cliaddr,servaddr;
listenfd=socket(AF_INET,SOCK_STREAM,0);
bzero(&servaddr,sizeof(servaddr));
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
servaddr.sin_port=htons(SERV_PORT);
bind(listenfd,(struct sockaddr *)&servaddr,sizeof(servaddr));
listen(listenfd,1);
maxfd=listenfd; maxi=-1;
for(i=0;i<FD_SETSIZE;i++) client[i]=-1;
FD_ZERO(&allset); FD_SET(listenfd,&allset);
for(;;)
```

```
{
rset=allset;
nready=select(maxfd+1,&rset,NULL,NULL,NULL);
if(FD_ISSET(listenfd,&rset))
{
clilen=sizeof(cliaddr);
connfd=accept(listenfd,(struct sockaddr *)&cliaddr,&clilen);
for(i=0;i<FD_SETSIZE;i++)if(client[i]<0)
{
client[i]=connfd;
break;
}
if(i==FD_SETSIZE)
{
printf("too many clients");
exit(0);
}
FD_SET(connfd,&allset); if(connfd>maxfd)
maxfd=connfd;
if(i>maxi)
maxi=i; if(--nready<=0)
continue;
}
for(i=0;i<=maxi;i++)
{
if((sockfd=client[i])<0)
continue; if(FD_ISSET(sockfd,&rset))
{
if((n=read(sockfd,line,MAXLINE))==0)
{
close(sockfd); FD_CLR(sockfd,&allset);
client[i]=-1;
}
else
{
```

```
printf("line received from client:%s\n",line);
for(j=0;line[j]!='\0';j++)
line[j]=toupper(line[j]);
write(sockfd,line,MAXLINE);
}
if(--nready<=0) break;
}
}
}
}
```

OUTPUT :



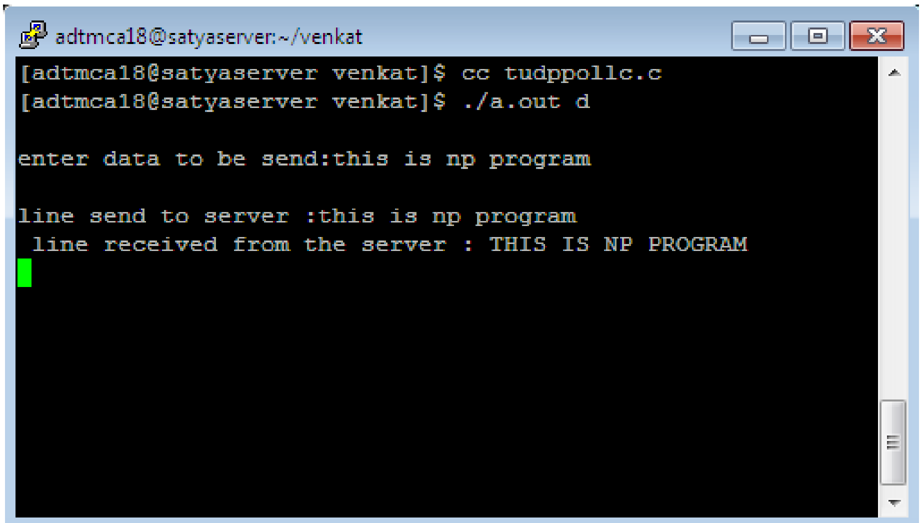
```
[adtmca18@satyaserver venkat]$ cc tudppolls.c
[adtmca18@satyaserver venkat]$ ./a.out
line received from client:this is np program
```

Client Program :

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<unistd.h>
#include<netinet/in.h>
#define MAXLINE 20
```

```
#define SERV_PORT 8114 main(int argc,char **argv)
{
int maxfdp1;
fd_set rset;
char sendline[MAXLINE],recvline[MAXLINE];
int sockfd;
struct sockaddr_in servaddr; if(argc!=2)
{
printf("usage tcpcli <ipaddress>"); return;
} sockfd=socket(AF_INET,SOCK_STREAM,0);
bzero(&servaddr,sizeof(servaddr));
servaddr.sin_family=AF_INET;
servaddr.sin_port=htons(SERV_PORT);
inet_pton(AF_INET,argv[1],&servaddr.sin_addr);
connect(sockfd,(struct sockaddr *)&servaddr,sizeof(servaddr));
printf("\nenter data to be send:");
while(fgets(sendline,MAXLINE,stdin)!=NULL)
{
write(sockfd,sendline,MAXLINE);
printf("\nline send to server :%s ",sendline);
read(sockfd,recvline,MAXLINE);
printf("line received from the server : %s",recvline);
}
exit(0);
}
```

OUTPUT :

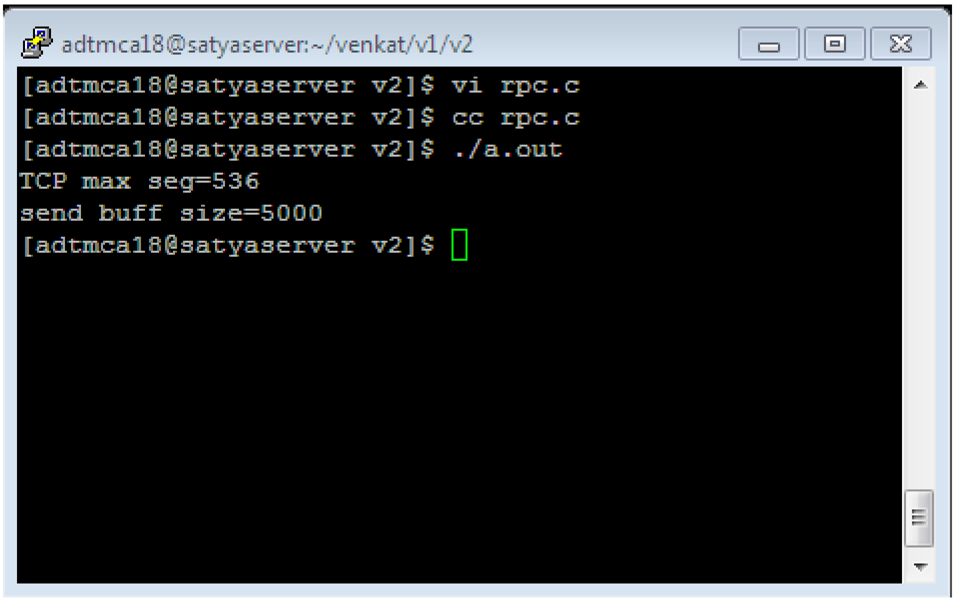


```
[adtmca18@satyaserver venkat]$ cc tudppollc.c
[adtmca18@satyaserver venkat]$ ./a.out d

enter data to be send:this is np program

line send to server :this is np program
 line received from the server : THIS IS NP PROGRAM
```

10. Design a RPC application to add and subtract a given pair of integers.

```
#include<stdio.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<string.h>

#include<netinet/in.h>

#include<netinet/tcp.h>

main()

{

int sockfd,maxseg,sendbuff,optlen;

sockfd=socket(AF_INET,SOCK_STREAM,0);

optlen=sizeof(maxseg);

if(getsockopt(sockfd,IPPROTO_TCP,TCP_MAXSEG,(char *)&maxseg,&optlen)<0)

printf("Max seg error");

else

printf("TCP max seg=%d\n",maxseg);

sendbuff=2500;

if(setsockopt(sockfd,SOL_SOCKET,SO_SNDBUF,(char*)&sendbuff,sizeof(sendbuff))<
0)
printf("set error");

optlen=sizeof(sendbuff);

getsockopt(sockfd,SOL_SOCKET,SO_SNDBUF,(char *)&sendbuff,&optlen);

printf("send buff size=%d\n",sendbuff);

}
```

OUTPUT :



```
[adtmca18@satyaserver v2]$ vi rpc.c
[adtmca18@satyaserver v2]$ cc rpc.c
[adtmca18@satyaserver v2]$ ./a.out
TCP max seg=536
send buff size=5000
[adtmca18@satyaserver v2]$
```