NETWORKS PROGRAMS

PART – A

1. **Simulate a three nodes point – to – point network with duplex links between them. Set the queue size and vary the bandwidth and find the number of packets dropped.**

prog1.tcl

```
set ns [new Simulator]
set nf [open prog1.nam w]
$ns namtrace-all $nf
set nd [open prog1.tr w]
$ns trace-all $nd

proc finish { } {
global ns nf nd
$ns flush-trace
close $nf
close $nd
exec nam prog1.nam &
exit 0
}

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]

$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 512Kb 10ms DropTail
$ns queue-limit $n1 $n2 5

set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
set sink [new Agent/Null]
$ns attach-agent $n2 $sink
$ns connect $udp0 $sink

$ns at 0.2 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
$ns at 5.0 "finish"
$ns run
```

prog1.awk

```
BEGIN {
dcount = 0;
rcount = 0;
}
{
event = $1;
if(event == "d")
{
dcount++;
}
if(event == "r")
{
rcount++;
}
}
END {
printf("The no.of packets dropped  : %d\n ",dcount);
printf("The no.of packets recieved : %d\n ",rcount);
}
```

2. **Simulate a four node point-to-point network with the links connected as follows: n0 – n2, n1 – n2 and n2 – n3. Apply TCP agent between n0-n3 and UDP between n1-n3. Apply relevant applications over TCP and UDP agents changing the parameter and determine the number of packets sent by TCP / UDP.**

prog2.tcl

```
set ns [new Simulator]
set nf [open prog2.nam w]
$ns namtrace-all $nf
set nd [open prog2.tr w]
$ns trace-all $nd

proc finish {} {
global ns nf nd
$ns flush-trace
close $nf
exec nam prog2.nam &
exit 0
}

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

$n0 label TCP
$n1 label UDP
$n3 label NULL-TCPSINK

$ns duplex-link $n0 $n2 1Mb 10ms DropTail
```

```
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail

set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0
$ns connect $tcp0 $sink0

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0

set udp0 [new Agent/UDP]
$ns attach-agent $n1 $udp0
set null0 [new Agent/Null]
$ns attach-agent $n3 $null0
$ns connect $udp0 $null0

set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

$ns at 0.2 "$cbr0 start"
$ns at 0.1 "$ftp0 start"
$ns at 4.5 "$cbr0 stop"
$ns at 4.4 "$ftp0 stop"

$ns at 5.0 "finish"
$ns run
```

prog2.awk

```
BEGIN {
ctcp=0;
cudp=0;
}
{
pkt=$5;
if(pkt=="cbr") { cudp++;}
if(pkt=="tcp") { ctcp++;}
}
END {
printf("No of packets sent\nTcp : %d\nUdp : %d\n",ctcp,cudp);
}
```

3. **Simulate the different types of Internet traffic such as FTP and TELNET over a network and analyze the throughput.**

prog3a.tcl

```
set ns [new Simulator]
set nf [open prog3a.nam w]
```

```
$ns namtrace-all $nf
set nd [open prog3a.tr w]
$ns trace-all $nd

proc finish {} {
global ns nf nd
$ns flush-trace
close $nf
close $nd
exec nam prog3a.nam &
exit 0
}

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]

$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail
$ns duplex-link $n3 $n4 1Mb 10ms DropTail

set tcp [new Agent/TCP]
set sink [new Agent/TCPSink]
$ns attach-agent $n0 $tcp
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp

$ns at 0.2 "$ftp start"
$ns at 4.5 "$ftp stop"
$ns at 5.0 "finish"
$ns run
```

prog3b.tcl

```
set ns [new Simulator]
set nf [open prog3b.nam w]
$ns namtrace-all $nf
set nd [open prog3b.tr w]
$ns trace-all $nd

proc finish {} {
global ns nf nd
$ns flush-trace
close $nf
close $nd
exec nam prog3b.nam &
exit 0
}
```

```
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]

$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail
$ns duplex-link $n3 $n4 1Mb 10ms DropTail

set tcp0 [new Agent/TCP]
set tcpsink0 [new Agent/TCPSink]
$ns attach-agent $n0 $tcp0
$ns attach-agent $n4 $tcpsink0
set telnet0 [new Application/Telnet]
$telnet0 attach-agent $tcp0
$ns connect $tcp0 $tcpsink0

$ns at 0.2 "$telnet0 start"
$ns at 4.5 "$telnet0 stop"
$ns at 5.8 "finish"
$ns run
```

prog3.awk

```
BEGIN {
sSize =0 ;
startTime=5.0;
stopTime=0.1;
Tput=0;
}
{
event=$1;
time=$2;
from=$3;
to=$4;
pkt=$5;
size=$6;
fid=$7;
src=$8;
dst=$9;
seqn=$10;
pid=$11;
if(event == "+") {
if(time < startTime) { startTime=time; }
}
if(event == "r") {
if(time >stopTime) { stopTime =time; }
sSize+=size;
}
Tput=(sSize/(stopTime-startTime))*(8/1000);
printf("%f\t%f\n",time,Tput);
}
```

```
END {
}
```

4. **Simulate the transmission of ping messages over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.**

prog4.tcl

```
set ns [new Simulator]
set nf [open prog4.nam w]
$ns namtrace-all $nf
set nd [open prog4.tr w]
$ns trace-all $nd

proc finish {} {
global ns nf nd
$ns flush-trace
close $nf
close $nd
exec nam prog4.nam &
exit 0
}

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]

$ns duplex-link $n1 $n0 1Mb 10ms DropTail
$ns duplex-link $n2 $n0 1Mb 10ms DropTail
$ns duplex-link $n3 $n0 1Mb 10ms DropTail
$ns duplex-link $n4 $n0 1Mb 10ms DropTail
$ns duplex-link $n5 $n0 1Mb 10ms DropTail
$ns duplex-link $n6 $n0 1Mb 10ms DropTail

Agent/Ping instproc recv {from rtt} {
$self instvar node_
puts "node [$node_ id] recieved ping answer from \
$from with round-trip-time $rtt ms."
}

set p1 [new Agent/Ping]
set p2 [new Agent/Ping]
set p3 [new Agent/Ping]
set p4 [new Agent/Ping]
set p5 [new Agent/Ping]
set p6 [new Agent/Ping]

$ns attach-agent $n1 $p1
```

```
$ns attach-agent $n2 $p2
$ns attach-agent $n3 $p3
$ns attach-agent $n4 $p4
$ns attach-agent $n5 $p5
$ns attach-agent $n6 $p6

$ns queue-limit $n0 $n4 3
$ns queue-limit $n0 $n5 2
$ns queue-limit $n0 $n6 2

$ns connect $p1 $p4
$ns connect $p2 $p5
$ns connect $p3 $p6

$ns at 0.2 "$p1 send"
$ns at 0.4 "$p2 send"
$ns at 0.6 "$p3 send"
$ns at 1.0 "$p4 send"
$ns at 1.2 "$p5 send"
$ns at 1.4 "$p6 send"
$ns at 2.0 "finish"
$ns run
```

prog4.awk

```
BEGIN {
count=0;
}
{
event=$1;
if(event=="d")
{
count++;
}
}
END {
printf("No of packets dropped : %d\n",count);
}
```

5. **Simulate an Ethernet LAN using n nodes (6-10), change error rate and data rate and compare throughput.**

prog5.tcl

```
set ns [new Simulator]
set nf [open prog5.nam w]
$ns namtrace-all $nf
set nd [open prog5.tr w]
$ns trace-all $nd

proc finish {} {
global ns nf nd
$ns flush-trace
```

```
close $nf
close $nd
exec nam prog5.nam &
exit 0
}

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]

$ns make-lan "$n0 $n1 $n2 $n3 $n4 $n5 $n6" 0.2Mb 40ms LL Queue/DropTail
Mac/802_3

set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n5 $sink
$ns connect $tcp $sink

set ftp [new Application/FTP]
$ftp attach-agent $tcp

$ns at 1.0 "$ftp start"
$ns at 5.0 "$ftp stop"
$ns at 5.5 "finish"
$ns run
```

prog5.awk

```
BEGIN {
sSize=0;
startTime = 5.0;
stopTime = 0.1;
Tput = 0;
}
{
event = $1;
time = $2;
from = $3;
to = $4;
pkt = $5;
size = $6;
fid = $7;
src = $8;
dst = $9;
seqn = $10;
pid = $11;
if (event == "+") {
if(time < startTime) {
startTime = time;
```

```
}
}
if (event == "r") {
if(time > stopTime) {
stopTime = time;
}
sSize+=size;
}
Tput = (sSize/(stopTime-startTime))*(8/1000);
printf("%f\t%.2f\n",time,Tput);
}
END {
}
```

6. **Simulate an Ethernet LAN using n nodes and set multiple traffic nodes and determine collision across different nodes.**

prog6.tcl

```
set ns [new Simulator]
set nf [open prog6.nam w]
$ns namtrace-all $nf
set nd [open prog6.tr w]
$ns trace-all $nd

$ns color 1 Blue
$ns color 2 Red

proc finish {} {
global ns nf nd
$ns flush-trace
close $nf
close $nd
exec nam prog6.nam &
exit 0
}

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]
set n8 [$ns node]

$ns duplex-link $n1 $n0 2Mb 10ms DropTail
$ns duplex-link $n2 $n0 2Mb 10ms DropTail
$ns duplex-link $n0 $n3 1Mb 20ms DropTail

$ns make-lan "$n3 $n4 $n5 $n6 $n7 $n8" 512Kb 40ms LL Queue/DropTail
```

```
$ns duplex-link-op $n1 $n0 orient right-down
$ns duplex-link-op $n2 $n0 orient right-up
$ns duplex-link-op $n0 $n3 orient right

set tcp [new Agent/TCP]
$ns attach-agent $n1 $tcp
set sink1 [new Agent/TCPSink]
$ns attach-agent $n4 $sink1
$ns connect $tcp $sink1
set sink2 [new Agent/TCPSink]
$ns attach-agent $n7 $sink2
$ns connect $tcp $sink2
$tcp set class_ 1
$tcp set packetSize_ 500

set ftp [new Application/FTP]
$ftp attach-agent $tcp

set udp [new Agent/UDP]
$ns attach-agent $n2 $udp
set null1 [new Agent/Null]
$ns attach-agent $n5 $null1
$ns connect $udp $null1
set null2 [new Agent/Null]
$ns attach-agent $n8 $null2
$ns connect $udp $null2
$udp set class_ 2

set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set packetSize_ 500
$cbr set rate_ 0.01Mb
$cbr set random_ false

$ns at 0.5 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 5.0 "$ftp stop"
$ns at 5.0 "$cbr stop"

$ns at 5.5 "finish"
$ns run
```

prog6.awk

```
BEGIN {
count=0;
count1=0;
}
{
event=$1;
if(event=="d")
{
count++;
```

```
}
if(event=="r")
{
count1++;
}
}
END {
printf("No of packets dropped and recieved : %d %d\n",count,count1);
}
```

7. **Simulate an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination.**

prog7.tcl

```
set ns [new Simulator]
set nf [open prog7.nam w]
$ns namtrace-all $nf
set nd [open prog7.tr w]
$ns trace-all $nd

$ns color 1 Blue
$ns color 2 Red

proc finish { } {
global ns nf nd
$ns flush-trace
close $nf
close $nd
exec nam prog7.nam &
exit 0
}

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]
set n8 [$ns node]

$n7 shape box
$n7 color Blue
```

```
$n8 shape hexagon
$n8 color Red

$ns duplex-link $n1 $n0 2Mb 10ms DropTail
$ns duplex-link $n2 $n0 2Mb 10ms DropTail
$ns duplex-link $n0 $n3 1Mb 20ms DropTail

$ns make-lan "$n3 $n4 $n5 $n6 $n7 $n8" 512Kb 40ms LL
Queue/DropTail Mac/802_3

$ns duplex-link-op $n1 $n0 orient right-down
$ns duplex-link-op $n2 $n0 orient right-up
$ns duplex-link-op $n0 $n3 orient right

$ns queue-limit $n0 $n3 20

set tcp1 [new Agent/TCP/Vegas]
$ns attach-agent $n1 $tcp1
set sink1 [new Agent/TCPSink]
$ns attach-agent $n7 $sink1
$ns connect $tcp1 $sink1
$tcp1 set class_ 1
$tcp1 set packetsize_ 55

set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1

set tfile [open cwnd.tr w]
$tcp1 attach $tfile
$tcp1 trace cwnd_

set tcp2 [new Agent/TCP/Reno]
$ns attach-agent $n2 $tcp2
set sink2 [new Agent/TCPSink]
$ns attach-agent $n8 $sink2
$ns connect $tcp2 $sink2
$tcp2 set class_ 2
$tcp2 set packetSize_ 55

set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2

set tfile2 [open cwnd2.tr w]
$tcp2 attach $tfile2
$tcp2 trace cwnd_

$ns at 0.5 "$ftp1 start"
```

```
$ns at 1.0 "$ftp2 start"
$ns at 5.0 "$ftp2 stop"
$ns at 5.0 "$ftp1 stop"

$ns at 5.5 "finish"
$ns run
```

prog7.awk

```
BEGIN {
}
{
if($6=="cwnd_") {
printf("%f\t%f\n",$1,$7);
}
}
END {
}
```

PART - B

1. **Write a program for error detecting code using CRC-CCITT (16- bits).**

```
#include<stdio.h>
#define MAX 25

int n=0,np=17;
int crc[MAX],temp[MAX],input[MAX];
int poly[17]={1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,1};

void genCRC(void);
void convert2bin(int);
void makeerror(int);

int main()
{
      int i, j, num, pos, ch, f=0;
      clrscr();
      printf("\n\nEnter the dataword(number)\n");
```

```
scanf("%d",&num);
convert2bin(num);
printf("The binary equivalent of %d is\n",num);
for(i=1;i<n;i++)
printf(" %d",input[i]);

for(i=n;i<n+np-1;i++)
input[i]=0;

genCRC();
printf("\nGenerated CRC is:\n");
for(i=n,j=0;i<n+np-1;i++)
{
     input[i]=temp[j++];
     printf(" %d",input[i]);
 }
 printf("\nTransmitted code word is:\n");
 for(i=1;i<n+np-1;i++)
 printf(" %d ",input[i]);

printf("\n\nDo you want to make an error(1/0): ");
scanf("%d",&ch);
switch(ch)
{
       case 1: printf("Enter the position\n");
            scanf("%d",&pos);
            if(pos>0 && pos<n+np-1)
               f=1;
            else
               printf("Invalid position\n\n");

            if(f)
            {
                makeerror(pos);
                printf("\nReceived code word is:\n");
                for(i=1;i<n+np-1;i++)
                printf(" %d",input[i]);

                genCRC();
                for(i=0;i<np;i++)
                if(temp[i]!=0)
                 break;
                 printf("\nMessage received has error\n\n");
            }
            break;
        case 0: printf("\nNO change in code word\n\n");
}
```

```
      getch();
      return(0);
}

void  genCRC()
{
     int i,j,k,s;
     for(i=0;i<np;i++)
     temp[i]=input[i];
     while(i<n+np)
     {
      k=0;j=0;
      if(temp[k]==poly[j])
       while(k<np)
        crc[k]=temp[k++]^poly[j++];
      else
       while(k<np)
         crc[k]=temp[k++]^0;
      for(s=1,j=0;s<np;s++)
       temp[j++]=crc[s];
      temp[j]=input[i++];
     }
}

void convert2bin(int num)
{
    int i=0;
    while(num!=0)
    {
      temp[i++]=num%2;
      num=num/2;
    }
    while(i>=0)
    input[n++]=temp[i--];
}

void makeerror(int pos)
{
     if(input[pos]==1)
        input[pos]=0;
      else
        input[pos]=1;
}
```

2. **Write a program for frame sorting technique used in buffers.**

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

struct header
{
      char data[20];
      int seqno;
} frame[20],temp;

main()
{
      int i,j,n;
      clrscr();
      printf("Enter the number of frames\n");
      scanf("%d",&n);

      printf("\nEnter the sequence number and the data of %d
frames\n",n);
      for(i=0;i<n;i++)
      scanf("%d%s",&frame[i].seqno,frame[i].data);

      for(i=0;i<n-1;i++)
      {
         for(j=i+1;j<n;j++)
         {
            if(frame[i].seqno>frame[j].seqno)
             {
                temp=frame[j];
                frame[j]=frame[i];
                frame[i]=temp;
             }
         }
      }
      printf("\n Frames after sorting are\n");
      for(i=0;i<n;i++)
      printf("seqno=%d\t data=%s\n", frame[i].seqno,
frame[i].data);
      getch();
      return;
}
```

3. **Write a program for distance vector algorithm to find suitable path for transmission.**

```
#include<stdio.h>
#define MAX 10
```

```
#define NOEDGE 999

struct node
{
     int t[MAX][5];
}nodes[MAX];

void init(int,int);
void input(int,int);
void update(int,int);
void display(int);

int main()
{
       int n,i,j;
       clrscr();
          printf("\n ENTER THE NUMBER OF NODES (i.e ROUTER) : ");
          scanf("%d",&n);
       for(i=1;i<=n;i++)
       {
              init(n,i);
              input(n,i);
       }
      printf("\n\n*************** BEFORE UPDATION, ROUTING TABLES
ARE*************************\n");
         display(n);
      for(j=1;j<=n;j++)
         update(n,j);
      printf("\n\n\n ******************** AFTER UPDATION, ROUTING
TABLES ARE***************\n");
         display(n);
         return 0;
}

void display(int n)
{
    int i,j;
    for(i=1;i<=n;i++)
    {
        printf("\n\n ROUTING TABLE OF NODE %d \n",i);
        printf("\tDESTINATION\tDISTANCE\tNEXT-HOP\tHOP-COUNT\n");

printf("-------------------------------------------------------------
--------------\n");
      for(j=1;j<=n;j++)
         {
              printf("\t    %d\t",nodes[i].t[j][1]);
```

```
            printf("\t    %d\t",nodes[i].t[j][2]);
            printf("\t    %d\t",nodes[i].t[j][3]);
             printf("\t    %d\t",nodes[i].t[j][4]);
             printf("\n");
      }
    }
    getch();
}

void init(int n,int x)
{
        int j;
        for(j=1;j<=n;j++)
        {
        if(j!=x)
           {
                 nodes[x].t[j][1]=j;
                 nodes[x].t[j][2]=NOEDGE;
                  nodes[x].t[j][3]=j;
                 nodes[x].t[j][4]=0;
        }
      }
       nodes[x].t[x][1]=x;
      nodes[x].t[x][2]=0;
      nodes[x].t[x][3]=x;
      nodes[x].t[x][4]=0;
}

void input(int n,int x)
{
        int i;
        printf("\n Enter distance to other nodes from %d\n",x);
        printf("Enter 999 to indicated no-edge\n");
        for(i=1;i<=n;i++)
            if(i!=x)
            {
                printf("\n enter dist to %d: ",i);
                scanf("%d",&nodes[x].t[i][2]);

                if(nodes[x].t[i][2]!=NOEDGE)
                {
                        nodes[x].t[i][3]=i;
                        nodes[x].t[i][4]=1;
                }
            }
}
```

```
void update(int n,int x)
{
        int i,j,z;
        for(i=1;i<=n;i++)
             if(nodes[x].t[i][2]!=NOEDGE && nodes[x].t[i][2]!=0)
             {
                 for(j=1;j<=n;j++)
                 {
                      z=nodes[x].t[i][2]+nodes[i].t[j][2];
                      if(nodes[x].t[j][2]>z)
                      {
                             nodes[x].t[j][2]=z;
                             if(nodes[x].t[i][4]==1)
                               nodes[x].t[j][3]=i;
                             else if(nodes[x].t[i][4]>1)
                              nodes[x].t[j][3]= nodes[x].t[i][3];

            nodes[x].t[j][4]= nodes[x].t[i][4]+ nodes[i].t[j][4];
                      }
                 }
              }
}
```

4. **Using TCP/IP sockets, write a client – server program to make the client send the file name and to make the server send back the contents of the requested file if present.**

/*Socket- Server*/

```
#include<stdio.h>
#include<string.h>
#include<fcntl.h>
#include<sys/types.h>
#include<sys/socket.h>
#define PORT_ID 50
struct sockaddr_in
{
short int sin_family;
unsigned short int sin_port;
unsigned long s_addr;
unsigned char sin_zero[8];
};
int main()
{
char buf[2000];
int fd1,n,fd2,size;
```

```
struct sockaddr_in s,s1;
printf("server is ready\n");
s.sin_family=AF_INET;
s.sin_port=htons(PORT_ID);
s.s_addr=inet_addr("127.0.0.1");
bzero(&(s.sin_zero),8);
fd1=socket(AF_INET,SOCK_STREAM,0);
if((bind(fd1,(struct sockaddr*)&s,sizeof(struct sockaddr)))==-1)
printf("Error.\n");
if((listen(fd1,5))==-1)
printf("Error\n");
size=sizeof(struct sockaddr);
printf("Waiting for client request\n");
fd2=accept(fd1,(struct sockaddr*)&s1,&size);
close(fd1);
size=recv(fd2,buf,sizeof(buf),0);
buf[size]='\0';
printf("File:%s\n",buf);
if((fd1=open(buf,O_RDONLY))!=-1)
while((n=read(fd1,buf,sizeof(buf)))>0)
send(fd2,buf,n,0);
else
send(fd2,"File not found",15,0);
close(fd1);
close(fd2);
printf("server terminated\n");
return 0;
}
```

/*Socket-Client*/

```
#include<stdio.h>
#include<string.h>
#include<sys/types.h>
#include<sys/socket.h>
#define P_ID 50
struct sockaddr_in
{
short int sin_family;
unsigned short int sin_port;
unsigned long s_addr;
unsigned char sin_zero[8];
};
int main()
{
char buf[2000];
int fd1,n;
struct sockaddr_in s,s1;
```

```
printf("Enter file request to server:");
scanf("%s",buf);
s.sin_family=AF_INET;
s.sin_port=htons(P_ID);
s.s_addr=inet_addr("127.0.0.1");
bzero(&(s.sin_zero),8);
fd1=socket(AF_INET,SOCK_STREAM,0);
if((connect(fd1,(struct sockaddr*)&s,sizeof(struct
sockaddr)))==-1)
printf("Error.\n");
send(fd1,buf,strlen(buf),0);
printf("\ncontents of file are\n\n");
while((n=recv(fd1,buf,sizeof(buf),0))>0)
{
buf[n]='\0';
printf("%s",buf);
}
printf("\n");
close(fd1);
return 0;
}
```

5. **Implement the above program using as message queues or FIFOs as IPC channels.**

/*FIFO-Server*/

```
#include<stdio.h>
#include<fcntl.h>
#include<sys/stat.h>

main()
{
      char buf[1000];
      int fd1,fd2,n,fd;
      mkfifo("LIFE",S_IFIFO|0777);
      printf("\n server started\n server is waiting");
      fd1=open("LIFE",O_RDONLY);
      n=read(fd1,buf,128);
      buf[n]='\0';
      close(fd1);
      fd2=open("LIFE",O_WRONLY);
      if((fd=open(buf,O_RDONLY))<0)
      {
            write(fd2,"file not found\n",16);
            printf("server termintated");
      }
```

```
      while((n=read(fd,buf,128))>0)
      write(fd2,buf,n);
      close(fd);
      close(fd2);
      printf("\n server terminates\n");
}
```

/*FIFO-Client*/

```
#include<stdio.h>
#include<fcntl.h>
#include<string.h>

main()
{
       char buf[1000];
       int fd1,fd2,n;
       fd1=open("LIFE",O_WRONLY);
      printf("\n enter the filename: ");
      scanf("%s",buf);
      write(fd1,buf,strlen(buf));
      close(fd1);
       fd2=open("LIFE",O_RDONLY);
      while((n=read(fd2,buf,128))>0)
      write(1,buf,n);
      close(fd2);
}
```

6. **Write a program for simple RSA algorithm to encrypt and decrypt the data.**

```
#include<stdio.h>
#include<math.h>
#include<string.h>
#define max 20

int len,n,e,p,q, CIPHER[max];

int main()
{
      int i,j,d,z;
      int PLAIN[max],sum[max],asc;
      char M[max];
      clrscr();

      printf("\n Enter the message to encrypt: ");
      for(len=0;len<max;len++)
```

```
{
      scanf("%c",&M[len]);
      if(M[len]=='\n')
      break;
}

printf("\n Entered string is\n");
for(i=0;i<len;i++)
printf(" %c ",M[i]);
   printf("\n Corresponding ASCII values are\n");
for(i=0;i<len;i++)
   printf(" %d ",M[i]);

do
{
printf("\n\n Enter 2 large unequal prime num p & q: ");
      scanf("%d%d",&p,&q);
   } while(p==q);

n=p*q;
z=(p-1)*(q-1);

e=1;
do
{
     e++;
} while( (gcd(e,z)!=1) && e<z );

printf("\n encryption key is %d ",e);
printf("\n public key (e,n): %d %d\n",e,n);

d=1;
   do
   {
        d++;
   } while( (e*d)%z!=1 && d<z );

printf("\n decryption key is %d \n",d);
   printf("\n\n private key (d,n): %d %d\n",d,n);


for(i=0;i<len;i++)
   CIPHER[i]=1;
   for(i=0;i<len;i++)
   {
            asc=M[i];
      for(j=1;j<=e;j++)
```

```
            CIPHER[i] = ((CIPHER[i]*asc) % n);
      }
      for(i=0;i<len;i++)
      printf("\n CIPHERTEXT for mesg %c is %d ",M[i],CIPHER[i]);

         printf("\n---------Decryption starts  now-------\n\n");

         for(i=0;i<len;i++)
      PLAIN[i]=1;
         for(i=0;i<len;i++)
            {

            sum[i]=1;
            for(j=1;j<=d;j++)
             PLAIN[i]=(PLAIN[i]*((sum[i]*CIPHER[i]) % n))%n;
      }
      for(i=0;i<len;i++)
printf("Number after decryption is %d & hence original Character
is %c \n",PLAIN[i],PLAIN[i]);
      getch();
      return 0;
}


int gcd(int a,int b)
{
       if (b==0) return a;
       else return (gcd(b,a%b));
}
```

7. **Write a program for Hamming code generation for error detection and correction.**

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

void genhamcode();
void makeerror();
void correcterror();

int h[12];

int main()
{
     int i,ch;
     clrscr();
```

```
     printf("\n enter the message in bits\n");
     for(i=1;i<12;i++)
          if(i==3||i==5||i==6||i==7||i==9||i==10||i==11)
             scanf("%d",&h[i]);
     genhamcode();

      printf("\n do you want to make error\n(0 or 1)\n");
      scanf("%d",&ch);
      if(ch)
      {
            makeerror();
             correcterror();
      }
      else
      printf("\n no error");
      getch();
      return(0);
 }

void genhamcode()
{
   int temp,i;
   temp=h[3]+h[5]+h[7]+h[9]+h[11];
    (temp%2!=0)?(h[1]=1):(h[1]=0);

    temp=h[3]+h[6]+h[7]+h[10]+h[11];
    (temp%2!=0)?(h[2]=1):(h[2]=0);

     temp=h[5]+h[6]+h[7];
    (temp%2!=0)?(h[4]=1):(h[4]=0);

     temp=h[9]+h[10]+h[11];
    (temp%2!=0)?(h[8]=1):(h[8]=0);

    printf("\n transmitted codeword is:\n");
    for(i=1;i<12;i++)
     printf(" %d ",h[i]);
  }

void makeerror()
{
  int pos,i;
  printf("\n enter the position you want to make error\n");
  scanf("%d",&pos);
  if(h[pos]==1)
     h[pos]=0;
  else
```

```
     h[pos]=1;
  printf("\n Error occured and the error codeword is\n");
  for(i=1;i<12;i++)
  printf(" %d ",h[i]);
}

void correcterror()
{
     int r1,r2,r4,r8,i,errpos;

     r1=(h[1]+h[3]+h[5]+h[7]+h[9]+h[11])%2;
     r2=(h[2]+h[3]+h[6]+h[7]+h[10]+h[11])%2;
     r4=(h[4]+h[5]+h[6]+h[7])%2;
     r8=(h[8]+h[9]+h[10]+h[11])%2;

     errpos=r8*8+r4*4+r2*2+r1*1;
     printf("\n Error occured in pos %d\n",errpos);

  printf("\n\n..... correction starts now........\n");
       if(h[errpos]==1)
           h[errpos]=0;
       else
           h[errpos]=1;

       printf("\n Original codeword is :");
       for(i=1;i<12;i++)
       printf(" %d ",h[i]);
}
```

8. **Write a program for congestion control using leaky bucket algorithm.**

```
#include<stdio.h>

int main()
 {
     int bcktsize, pkt[25], i, j, iter, rate, line, total=0;
     clrscr();

     printf("Enter the bucket size and output rate :");
     scanf("%d%d",&bcktsize,&rate);

     printf("Enter the number of input lines\n");
        scanf("%d",&line);
```

```
      printf("Enter input packet rate of %d lines\n",line);
      for(i=0;i<line;i++)
      scanf("%d",&pkt[i]);

      printf("Enter the number of iterations\n");
       scanf("%d",&iter);

      for(i=0;i<iter;i++)
         {
        printf("\nIteration %d\n",i+1);
           for(j=0;j<line;j++)
        {
             total+=pkt[j];
             if(total<=bcktsize)
    printf("\nInput from line %d with rate %d is added to the
bucket\nNumber of packets in bucket is %d\n",j+1,pkt[j],total);
           else
           {
              total-=pkt[j];
   printf("\nInput from line %d with rate %d is  thrown out of
bucket\nNumber of packets in bucket is %d\n",j+1,pkt[j],total);
    }
   }
  total-=rate;
  printf("\n---------------------------------\n");
  printf("\npacket sent to output line at rate %d \nNumber of
packets in bucket is %d\n",rate,total);
   printf("\n---------------------------------\n");

      }
      getch();
      return 0;
 }
```