1. **3 nodes point to point network**

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

proc finish {} {
global ns nf tf $ns flush-trace
close $nf
close $tf
exec nam out.nam &
exit 0
}
set n0 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

$ns duplex-link $n0 $n2 200Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 1000ms DropTail
$ns queue-limit $n0 $n2 10
```

```
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 null0 [new Agent/Null]
$ns attach-agent $n3 $null0
$ns connect $udp0 $null0

$ns at 0.1 "$cbr0 start"
$ns at 1.0 "finish"
$ns run

//AWK file

BEGIN
{
c=0;
}
{
if($1=="d")
{
```

```
    C++;
    printf("%s\t%s\n",$5,$11);
    }
    }
    END{ printf("The number of packets dropped = %d\n",c);
}
```

2. bit stuffing and character stuffing

Bit stuffing

```
#include<stdio.h>
#include<string.h>
void main() { char ch,array[50]={"01111110"},recd_array[50];
int counter=0,i=8,j,k;
printf("Enter the original data stream for bit stuffing:\n");
while((ch=getchar())!='\n')
{ if(ch=='1')
++counter;
else
counter=0; array[i++]=ch;
if(counter==5)
{
array[i++]='0';
counter=0;
```

```
}
}
strcat(array,"01111110");
printf("\n The stuffed data stream is:\n");
for(j=0;j<i+8;++j)
printf("%c",array[j]);
counter=0;
printf("\n The destuffed data stream is:\n");
for(j=8,k=0;j<i+8;++j)
{
if(array[j]=='1')
++counter;
else
counter=0;
recd_array[k++]=array[j];
if(counter==6)
break;
else if(counter==5 && array[j+1]=='0')
{
++j;
counter=0;
}
}
for(j=0;j<=k-strlen("01111110");++j)
```

```
printf("%c",recd_array[j]);
}
```

Character stuffing

```
#include<stdio.h>
#include<string.h>
main()
{
char a[50],s1[100]={"DLE"},s2[10];
int i,j,l;
printf("\n character stuffing and unstuffing program\n");
printf("\n @ SENDER --\n");
printf("\n enter the message to be sent:\n");
gets(a);
l=strlen(a); //string length
for(i=0;i<l;i++)



{
if((a[i]=='D'&&a[i+1]=='L'&&a[i+2]=='E')||
(a[i]=='E'&&a[i+1]=='S'&&a[i+2]=='C'))
{
for(j=l+3;j>=i+3;j--)
```

```
{ a[j]=a[j-3]; //shifted data three position to the right
}
l=l+3;
a[i]='E';
a[i+1]='S';
a[i+2]='C';
i=i+5;
}
}
printf("\n message after character stuffing:\n");
printf("%s\n",a);
strcpy(s2,s1);
strcat(s1,a);
strcat(s1,s2);
printf("\n the transmitted frame:\n");
printf("%s\n",s1);
printf("\n--------------------------------\n");
printf("\n @RECEIVER --\n");
l=strlen(s1); //lenght of flag+stuffed_message+flag
s1[l-3]='\0'; //remove end delimiter(flag)
l=strlen(s1);
for(i=0;i<l;i++)
s1[i]=s1[i+3]; //remove start delimiter(flag)
l=strlen(s1);
```

```
printf("\nmessage after flag removal at receiver:\n");
printf("%s\n",s1);
for(i=0;i<l;i++)
{
if(s1[i]=='E'&&s1[i+1]=='S'&&s1[i+2]=='C')
{
for(j=i;j<=l;j++)
{ s1[j]=s1[j+3]; //shifted data three position to the left
}
l=l-3;
i=i+2;
}
}
printf("\nmessage after unstuffing:\n");
printf("%s\n",s1);
}
}
```

3. CRC CODE AND ERROR DETECTION

```
#include<stdio.h>

int a[100],b[100],i,j,len,k,count=0;
```

```
int gp[]={1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,1,};

int main()
{
void div():
printf("\nEnter the length of Data Frame:");
scanf("%d",&len);
printf("\nEnter the Message:");
for(i=0;i<len;i++)
scanf("%d",&a[i]);
for(i=0;i<16;i++)
a[len++]=0;
for(i=0;i<len;i++)
b[i]=a[i];
k= len-16;
div();
for(i=0;i<len;i++)
b[i]=b[i]^a[i];
printf("\nData to be transmitted: ");
for(i=0;i<len;i++)
printf("%2d",b[i]);
printf("\n\nEnter the Reveived Data: ");
for(i=0;i<len;i++)
scanf("%d",&a[i]);
div();
for(i=0;i<len;i++)
if(a[i]!=0)
{
printf("\nERROR in Recived Data");
return 0;
}
printf("\nData Recived is ERROR FREE");
}
```

```
void div()
{

for(i=0;i<k;i++)
{
if(a[i]=gp[0])
{
for(j=i;j<17+i;j++)
a[j]=a[i]^gp[count++];
}
count=0;
}
}
```

4. GO BACK -N MECHENASIM

```
#include<stdio.h>
void main()
{
char sender[50],receiver[50];
int i,winsize;
printf("\n ENTER THE WINDOWS SIZE: ");
scanf("%d",&winsize);
printf("in SENDER WINDOW IS EXPANDED TO STORE MESSAGE \n");
printf("\n ENTER THE DATA TO BE SENT: ");
fflush(stdin);
scanf("%s",sender);
for(i=0;i<winsize;i++)
```

```
receiver[i]=sender[i];
printf("\n MESSAGE SEND BY THE SENDER:\n");
scanf("%s",sender);
printf("in WINDOW SIZE OF RECEIVER IS EXPANDED\n");
printf("\n ACKNOWLEDGEMENT FROM RECEIVER \n");
for(i=0;i<winsize;i++);
printf("\n ACK:%d",i);
printf("\n MESSAGE RECEIVED BY RECEIVER IS :");
scanf("%s",reciver);
printf("\n WINDOW SIZE OF RECEIVER IS SHRINKED \n");
}
```

5. Dijiskitra

```
#include<stdio.h>
void main()
{
int path[5][5], i, j, min, a[5][5], p, st=1,ed=5,stp,edp,t[5],index;
printf("Enter the cost matrix\n");
for(i=1;i<=5;i++)
for(j=1;j<=5;j++)
scanf("%d",&a[i][j]);
printf("Enter the paths\n");
scanf("%d",&p);
printf("Enter possible paths\n");
for(i=1;i<=p;i++)
for(j=1;j<=5;j++)
scanf("%d",&path[i][j]);
```

```
for(i=1;i<=p;i++)
{
t[i]=0;
stp=st;
for(j=1;j<=5;j++)
{
edp=path[i][j+1];
t[i]=t[i]+a[stp][edp];
if(edp==ed)
break;
else
stp=edp;
}
}
min=t[st];index=st;
for(i=1;i<=p;i++)
{
if(min>t[i])
{
min=t[i];
index=i;
}
}
printf("Minimum cost %d",min);
printf("\n Minimum cost path ");
for(i=1;i<=5;i++)
{
printf("--> %d",path[index][i]);
if(path[index][i]==ed)
break;
}
}
```

6. **Encrypit and decript**

```
#include <stdio.h>
int main()
{
 int i, x;
 char str[100];
 printf("\nPlease enter a string:\t");
 gets(str);
 printf("\nPlease choose following options:\n");
 printf("1 = Encrypt the string.\n");
 printf("2 = Decrypt the string.\n");
 scanf("%d", &x);
 switch(x)
 {
 case 1:
 for(i = 0; (i < 100 && str[i] != '\0'); i++)
 str[i] = str[i] + 3; //the key for encryption is 3 that is added to
ASCII value
 printf("\nEncrypted string: %s\n", str);
 break;
 case 2:
 for(i = 0; (i < 100 && str[i] != '\0'); i++)
 str[i] = str[i] - 3; //the key for encryption is 3 that is subtracted to
ASCII value
 printf("\nDecrypted string: %s\n", str);
 break;
 default:
 printf("\nError\n");
 }
 return 0;
}
```

7. **Working with nodes n0,n1,n2,n3,n4**

```
set ns [new Simulator]
set namfile [open 7.nam w]
$ns namtrace-all $namfile
set tracefile [open 7.tr w]
$ns trace-all $tracefile

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

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

set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp


set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n2 $null
$ns connect $udp $null
```

```
set cbr [new Application/Traffic/CBR]
$cbr set packet_size_  500
$cbr set interval_ 0.005
$cbr attach-agent $udp
$ns at 0.0 “$cbr start”
$ns at 0.0 “$ftp start”
$ns at 9.0 “$cbr stop”
$ns at 9.0 “$ftp stop”

proc finish {} {
global ns namfile tracefile
$ns flush-trace
close $namfile
close $tracefile
exec nam 7.nam &amp;amp;
exit 0
}
$ns at 10.0 “finish”
$ns run
```

8. LAN PACKETS

```
set ns [new Simulator]
set namfile [open 8b.nam w]
$ns namtrace-all $namfile
set tracefile [open 8b.tr w]
$ns trace-all $tracefile
set n0 [$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
$ns make-lan  “$n2 $n3 $n4” 100Mb 1ms LL
Queue/DropTail Mac/802_3 channel phy/wiredPhy
Set errmodel [new ErrorModel]
$errmodel set rate_ 0.2
$errmodel ranvar [new Randomvariable/uniform]
$errmodel drop-target[new Agent/Null]
$ns lossmodel $errmodel $n1 $n2
set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
Set filesize [expr 4*1024*1024]
$ns at 0.0 “$ftp send $filesize”
proc finish { } {
global ns namfile tracefile
$ns flush-trace
```

```
close $namfile
close $tracefile
set awkcode {

BEGIN
}
{
{
if($1==”d” && $5==”tcp” && $6 >1460)
{
Count_packets++;
Printf $2,count_packets >> “8b.data”
}
}
END{}
}
exec awk $aekcode 8b.tr
exec nam 8b.nam &
exec xgraph –bb -tk -x time -y Dropped-packets
8b.data -bg white &
exit 0
}
$ns at 100.0 “finish”
$ns run
```