```
Week 4(CRC)
                                                   if(temp[i]==kev[i])
                                                              rem[j-1]='0';
 #include <stdio.h>
                                                     else
 #include <string.h>
                                                              rem[j-1]='1';
 void main() {
        int i, j, keylen, msglen, l=0;
                                              rem[keylen1]=input[i+keylen];
        char
 input[100],key[30],temp[30],quot
                                                             strcpy(temp,rem);
 [100],rem[30],key1[30],fdata[60];
                                                     }
        printf("Enter Data: ");
                                                     strcpy(rem,temp);
        gets(input);
                                                     printf("\nQuotient is ");
        printf("Enter Divisor: ");
                                                     for (i=0;i<msglen;i++)
        gets(key);
                                                     printf("%c",quot[i]);
        keylen=strlen(key);
                                                     printf("\nRemainder is ");
        msglen=strlen(input);
                                                     for (i=0;i<keylen-1;i++)
        strcpy(key1,key);
                                                     printf("%c",rem[i]);
        for (i=0;i< keylen-1;i++) {
                                                     for (i=0;i<msglen;i++)
                                               {
        input[msglen+i]='0';
                                                fdata[l]=input[i];
                                                |++;
        for (i=0;i<keylen;i++)
                                               }
        temp[i]=input[i];
                                                     for (i=0;i<keylen-1;i++)
        for (i=0;i<msglen;i++) {
                                               {
                 quot[i]=temp[0];
                                                fdata[l]=rem[i];
                 if(quot[i]=='0')
                                                l++;
        for (j=0;j<keylen;j++)
                                               }
                 key[i]='0'; else
                                               printf("\nFinal data is: ");
         for (j=0;j<keylen;j++)
                                               for(i=0;i<l;i++)
                 key[i]=key1[i];
                                                     printf("%c",fdata[i])
        for (j=keylen-1;j>0;j--) {
                                              }
```

```
for(i=0;i<c 1;i++)
Week 4 (Hamming)
                                                   printf("%d",code[i]);
#include <stdio.h>
                                                   printf("\n");
#include <math.h>
                                                   printf("Please enter the
int input[32];
                                            received Code Word:\n");
int code[32];
                                                   for(i=0;i<c 1;i++)
int ham calc(int,int);
void main()
                                                   scanf("%d",&code[i]);
     int n,i,p n = 0,c l,j,k;
       printf("Please enter the
                                                   int error pos = 0;
length of the Data Word: ");
                                                   for(i=0;i
       scanf("%d",&n);
       printf("Please enter the
                                                   int position = (int)pow(2,i);
Data Word:\n");
                                            int value = ham calc(position,c l);
       for(i=0;i<n;i++)
                                                           if(value!=0)
{ scanf("%d",&input[i]); }
                                                   error pos+=position;
i=0;
                                                   if(error pos == 1)
       while(n>(int)pow(2,i)-(i+1))
                                                   printf("The received Code
       { p n++;
                                            Word is correct.\n");
               i++:}
                                                   else
       c = p + n;
                                                      printf("Error at bit
       i=k=0;
                                            position: %d\n",error_pos);
       for(i=0;i<c 1;i++)
                                            int ham calc(int position, int c l)
       if(i==((int)pow(2,k)-1))
               { code[i]=0;
                                                   int count=0,i,j;
                       k++;
                                                   i=position-1;
                                                   while(i<c |)
               else
                                                   { for(j=i;j<i+position;j++)
               { code[i]=input[j];
                       j++;
                                                           if(code[j] == 1)
               } }
                                                                   count++;
       for(i=0;i<p_n;i++)
                                                           i=i+2*position;
       int position = (int)pow(2,i);
int value = ham calc(position,c l);
                                                   if(count%2 == 0)
       code[position-1]=value;
                                                           return 0;
                                                   else
       printf("\nThe calculated
                                                           return 1; }
Code Word is: ");
```

#PROGRAM: CSMA/CA	set sink [new
set ns [new Simulator]	Agent/TCPSink/DelAck]
\$ns color 1 Blue	\$ns attach-agent \$n4 \$sink
\$ns color 2 Red	\$ns connect \$tcp \$sink
set file1 [open out.tr w]	\$tcp set fid_1
set winfile [open WinFile w]	\$tcp set window_8000
\$ns trace-all \$file1	\$tcp set packetSize_552
set file2 [open out.nam w]	set ftp [new Application/FTP]
\$ns namtrace-all \$file2	\$ftp attach-agent \$tcp
proc finish {} {	\$ftp set type_FTP
global ns file1 file2	set udp [new Agent/UDP]
\$ns flush-trace	\$ns attach-agent \$n1 \$udp
close \$file1	set null [new Agent/Null]
close \$file2	\$ns attach-agent \$n5 \$null
exec nam out.nam &	\$ns connect \$udp \$null
exit 0	\$udp set fid_ 2
}	set cbr [new
set n0 [\$ns node]	Application/Traffic/CBR]
set n1 [\$ns node]	\$cbr attach-agent \$udp
set n2 [\$ns node]	\$cbr set type_ CBR
set n3 [\$ns node]	\$cbr set packet_size_ 1000
set n4 [\$ns node]	\$cbr set rate_ 0.01mb
set n5 [\$ns node]	\$cbr set random_false
\$n1 color red	\$ns at 0.1 "\$cbr start"
\$n1 shape box	\$ns at 1.0 "\$ftp start"
\$ns duplex-link \$n0 \$n2 2Mb 10ms	\$ns at 124.0 "\$ftp stop"
DropTail	<pre>proc plotWindow {tcpSource file} {</pre>
\$ns duplex-link \$n1 \$n2 2Mb 10ms	global ns
DropTail	set time 0.1
\$ns simplex-link \$n2 \$n3 0.3Mb	set now [\$ns now]
100ms DropTail	<pre>set cwnd [\$tcpSource set cwnd_]</pre>
\$ns simplex-link \$n3 \$n2 0.3Mb	<pre>set wnd [\$tcpSource set window_]</pre>
100ms DropTail	puts \$file "\$now \$cwnd"
set lan [\$ns newLan "\$n3 \$n4 \$n5"	\$ns at [expr \$now+\$time]
0.5Mb 40ms LL Queue/DropTail	"plotWindow \$tcpSource \$file" }
MAC/Csma/Ca Channel]	\$ns at 0.1 "plotWindow \$tcp
Setup a TCP connection	\$winfile"
set tcp [new Agent/TCP/Newreno]	\$ns at 5 "\$ns trace-annotate
\$ns attach-agent \$n0 \$tcp	\"packet drop\""
•	\$ns at 125.0 "finish" \$ns run

#PROGRAIVI: CSIVIA/CD	şns connect ştcp şsink
set ns [new Simulator]	\$tcp set fid_1
\$ns color 1 Blue	\$tcp set window_8000
\$ns color 2 Red	\$tcp set packetSize_ 552
set file1 [open out.tr w]	set ftp [new Application/FTP]
set winfile [open WinFile w]	\$ftp attach-agent \$tcp
\$ns trace-all \$file1	\$ftp set type_ FTP
set file2 [open out.nam w]	set udp [new Agent/UDP]
\$ns namtrace-all \$file2	\$ns attach-agent \$n1 \$udp
proc finish {} {	set null [new Agent/Null]
global ns file1 file2	\$ns attach-agent \$n5 \$null
\$ns flush-trace	\$ns connect \$udp \$null
close \$file1	\$udp set fid_2
close \$file2	set cbr [new
exec nam out.nam &	Application/Traffic/CBR]
exit 0	\$cbr attach-agent \$udp
}	\$cbr set type_ CBR
set n0 [\$ns node]	\$cbr set packet_size_ 1000
set n1 [\$ns node]	\$cbr set rate_0.01mb
set n2 [\$ns node]	\$cbr set random_false
set n3 [\$ns node]	\$ns at 0.1 "\$cbr start"
set n4 [\$ns node]	\$ns at 1.0 "\$ftp start"
set n5 [\$ns node]	\$ns at 124.0 "\$ftp stop"
\$n1 color red	\$ns at 124.5 "\$cbr stop"
\$ns duplex-link \$n0 \$n2 2Mb 10ms	proc plotWindow {tcpSource file} {
DropTail	global ns
\$ns duplex-link \$n1 \$n2 2Mb 10ms	set time 0.1
DropTail	set now [\$ns now]
\$ns simplex-link \$n2 \$n3 0.3Mb	set cwnd [\$tcpSource set cwnd_]
100ms DropTail	set wnd [\$tcpSource set window_]
\$ns simplex-link \$n3 \$n2 0.3Mb	puts \$file "\$now \$cwnd"
100ms DropTail	\$ns at [expr \$now+\$time]
set lan [\$ns newLan "\$n3 \$n4 \$n5"	"plotWindow \$tcpSource \$file" }
0.5Mb 40ms LL Queue/DropTail	\$ns at 0.1 "plotWindow \$tcp
MAC/Csma/Ca Channel]	\$winfile"
set tcp [new Agent/TCP/Newreno]	\$ns at 5 "\$ns trace-annotate
\$ns attach-agent \$n0 \$tcp	\"packet drop\""
set sink [new	\$ns at 125.0 "finish"
Agent/TCPSink/DelAck]	\$ns run
\$ns attach-agent \$n4 \$sink	

```
if(frameQ>0 && cansend==-1)
#stop and wait
include<stdio.h>
                                               printf("FRAME IN Q FOR
      int timer=0, wait for ack=-
                                         TRANSMISSION AT TIME=%d\n",i);
                                         if(frameQ>0)
1,frameQ=0,cansend=1,t=0;
      main()
                                         printf("frameQ=%d\n",frameQ);
      {
                                         printf("i=%d t=%d\n",i,t);
          int i,j,k;
          int frame[5];
                                         printf("value in
                                         frame=%d\n",frame[t]);
          //clrscr();
          printf("enter the time
                                               return 0;
                                          //
when data frame will be ready\n");
          for(j=0;j<3;j++)
                                         recv(inti)
                                         { printf("timer recvr=%d\n",timer);
              sender(i, &frame);
                                                        if(timer>0)
              recv(i);
                                                        {
          }
                                                            timer++;
      }
              {
                                                            if(timer==3)
                   wait for ack++;
                if(wait for ack==3)
                                                                 printf("
                                                      FRAME ARRIVED AT
                                         TIME=%d\n'',i);
              if(i==frame[t])
                                         wait for ack=0;
              {
                   frameQ++;
                                                                 timer=0;
                                                            }
                   t++;
              }
                                                            else
              if(frameQ==0)
                                                                 printf("
                   printf("NO
                                                      WAITING FOR FRAME
FRAME TO SEND at time=%d \n",i);
                                         AT TIME %d\n",i);
              if(frameQ>0 &&
                                               //
                                                                 return 0;
cansend==1)
                                                    }
              {
                   printf("FRAME
SEND AT TIME=%d\n",i);
                   cansend=-1;
                   frameQ--;
                   timer++;
                   printf("timer in
sender=%d\n",timer); }
```

SAMPLETOPOLOGY	exec nam out.nam &
set ns [new Simulator]	exit 0
set nf [open out.nam w]	}
\$ns namtrace-all \$nf	set n0 [\$ns node]
proc finish {} {	set n1 [\$ns node]
global ns nf	set n2 [\$ns node]
\$ns flush-trace	set n3 [\$ns node]
close \$nf	\$ns duplex-link \$n0 \$n1 1Mb 10ms
exec nam out.nam &	DropTail
exit 0	\$ns duplex-link \$n1 \$n2 1Mb 10ms
}	DropTail
set n0 [\$ns node]	\$ns duplex-link \$n2 \$n3 1Mb 10ms
set n1 [\$ns node]	DropTail
\$ns duplex-link \$n0 \$n1 1Mb 10ms	\$ns duplex-link-op \$n0 \$n1 orient
DropTail	right
set udp0 [new Agent/UDP]	\$ns duplex-link-op \$n1 \$n2 orient
\$ns attach-agent \$n0 \$udp0	right
set cbr0 [new	\$ns duplex-link-op \$n2 \$n3 orient
Application/Traffic/CBR]	right
\$cbr0 set packetSize_500	set udp0 [new Agent/UDP]
\$cbr0 set interval_ 0.005	\$udp0 set class_1
\$cbr0 attach-agent \$udp0	\$ns attach-agent \$n0 \$udp0
set null0 [new Agent/Null]	set cbr0 [new
\$ns attach-agent \$n1 \$null0	Application/Traffic/CBR]
\$ns connect \$udp0 \$null0	\$cbr0 set package_500
\$ns at 0.5 "\$cbr0 start"	\$cbr0 set interval_0.005
\$ns at 2.5 "\$cbr0 stop"	\$cbr0 attach-agent \$udp0
\$ns at 3.0 "finish"	set null0 [new Agent/Null]
\$ns run	\$ns attach-agent \$n2 \$null0
	\$ns connect \$udp0 \$null0
	\$ns at 0.5 "\$cbr0 start"
BUS TOPOLOGY	\$ns at 4.5 "\$cbr0 stop"
set ns [new Simulator]	\$ns at 5.0 "finish"
\$ns color 1 Blue	\$ns run
set nf [open out.nam w]	
\$ns namtrace-all \$nf	
proc finish {} {	
global ns nf	
\$ns flush-trace	
close \$nf	

RING TOPOLOGY	\$ns duplex-link-op \$n4 \$n5 orient
set ns [new Simulator]	left
\$ns color 1 Blue	\$ns duplex-link-op \$n5 \$n0 orient
\$ns color 2 Red	left-up
\$ns color 3 Green	\$ns duplex-link-op \$n0 \$n1
\$ns color 4 Black	queuePos 0.0
set nf [open out.nam w]	set udp0 [new Agent/UDP]
\$ns namtrace-all \$nf	\$udp0 set class_1
proc finish {} {	\$ns attach-agent \$n0 \$udp0
global ns nf	set cbr0 [new
\$ns flush-trace	Application/Traffic/CBR]
close \$nf	\$cbr0 set packetsize_500
exec nam out.nam &	\$cbr0 set interval_ 0.005
exit 0 }	\$cbr0 attach-agent \$udp0
set n0 [\$ns node]	set udp1 [new Agent/UDP]
set n1 [\$ns node]	\$udp1 set class_2
set n2 [\$ns node]	\$ns attach-agent \$n1 \$udp1
set n3 [\$ns node]	set cbr1 [new
set n4 [\$ns node]	Application/Traffic/CBR]
set n5 [\$ns node]	\$cbr1 set packetsize_500
\$ns duplex-link \$n0 \$n1 1Mb 10ms	\$cbr1 set interval_ 0.005
DropTail	\$cbr1 attach-agent \$udp1
\$ns duplex-link \$n1 \$n2 1Mb 10ms	set udp2 [new Agent/UDP]
DropTail	\$udp2 set class_3
\$ns duplex-link \$n2 \$n3 1Mb 10ms	\$ns attach-agent \$n2 \$udp2
DropTail	set cbr2 [new
\$ns duplex-link \$n3 \$n4 1Mb 10ms	Application/Traffic/CBR]
DropTail	\$cbr2 set packetsize_500
\$ns duplex-link \$n4 \$n5 1Mb 10ms	\$cbr2 set interval_ 0.005
DropTail	\$cbr2 attach-agent \$udp2
\$ns duplex-link \$n5 \$n0 1Mb 10ms	set udp3 [new Agent/UDP]
DropTail	\$udp3 set class_4
\$ns duplex-link-op \$n0 \$n1 orient	\$ns attach-agent \$n3 \$udp3
right-up	set cbr3 [new
\$ns duplex-link-op \$n1 \$n2 orient	Application/Traffic/CBR]
right	\$cbr3 set packetsize_500
\$ns duplex-link-op \$n2 \$n3 orient	\$cbr3 set interval_ 0.005
right-down	\$cbr3 attach-agent \$udp3
\$ns duplex-link-op \$n3 \$n4 orient	set udp4 [new Agent/UDP]
left-down	\$udp4 set class_5

\$ns attach-agent \$n4 \$udp4	set n3 [\$ns node]
set cbr4 [new	\$ns duplex-link \$n0 \$n2 1Mb 10ms
Application/Traffic/CBR]	DropTail
\$cbr4 set packetsize_ 500	\$ns duplex-link \$n1 \$n2 1Mb 10ms
\$cbr4 set interval_ 0.005	DropTail
\$cbr4 attach-agent \$udp4	\$ns duplex-link \$n3 \$n2 1Mb 10ms
set null0 [new Agent/Null]	DropTail
\$ns attach-agent \$n5 \$null0	\$ns queue-limit \$n2 \$n3 10
\$ns connect \$udp0 \$null0	\$ns duplex-link-op \$n0 \$n2 orient
\$ns connect \$udp1 \$null0	right-up
\$ns connect \$udp2 \$null0	\$ns duplex-link-op \$n1 \$n2 orient
\$ns connect \$udp3 \$null0	right-down
\$ns connect \$udp4 \$null0	\$ns duplex-link-op \$n2 \$n3 orient
\$ns at 0.1 "\$cbr0 start"	right
\$ns at 0.5 "\$cbr1 start"	\$ns duplex-link-op \$n2 \$n3
\$ns at 1.0 "\$cbr2 start"	queuePos 0.5
\$ns at 1.5 "\$cbr3 start"	set udp0 [new Agent/UDP]
\$ns at 2.0 "\$cbr4 start"	\$udp0 set class_1
\$ns at 3.5 "\$cbr4 stop"	\$ns attach-agent \$n0 \$udp0
\$ns at 4.0 "\$cbr3 stop"	set cbr0 [new
\$ns at 4.5 "\$cbr2 stop"	Application/Traffic/CBR]
\$ns at 5.0 "\$cbr1 stop"	\$cbr0 set packetsize_500
\$ns at 5.5 "\$cbr0 stop"	\$cbr0 set interval_ 0.005
\$ns at 6.0 "finish"	\$cbr0 attach-agent \$udp0
\$ns run	set udp1 [new Agent/UDP]
STAR TOPOLOGY	\$udp1 set class_2
set ns [new Simulator]	\$ns attach-agent \$n1 \$udp1
\$ns color 1 Blue	set cbr1 [new
\$ns color 2 Red	Application/Traffic/CBR]
set nf [open out.nam w]	\$cbr1 set packetsize_500
\$ns namtrace-all \$nf	\$cbr1 set interval_0.005
proc finish {} {	\$cbr1 attach-agent \$udp1
global ns nf	set null0 [new Agent/Null]
\$ns flush-trace	\$ns attach-agent \$n3 \$null0
close \$nf	\$ns connect \$udp0 \$null0
exec nam out.nam &	\$ns connect \$udp1 \$null0
exit 0 }	\$ns at 0.5 "\$cbr0 start"
set n0 [\$ns node]	\$ns at 1.0 "\$cbr1 start"
set n1 [\$ns node]	\$ns at 4.0 "\$cbr1 stop"
set n2 [\$ns node]	\$ns at 4.5 "\$cbr0 stop"

\$ns at 5.0 "finish"	\$ns duplex-link-op \$n0 \$n4 orient
\$ns run	right
MESH TOPOLOGY	\$ns duplex-link-op \$n1 \$n0 orient
set ns [new Simulator]	right-down
\$ns color 1 Red	\$ns duplex-link-op \$n1 \$n2 orient
\$ns color 2 Blue	right-up
set nf [open out.nam w]	\$ns duplex-link-op \$n2 \$n3 orient
\$ns namtrace-all \$nf	right-down
proc finish {} {	\$ns duplex-link-op \$n3 \$n4 orient
global ns nf	left-down
\$ns flush-trace	\$ns duplex-link-op \$n3 \$n2 orient
close \$nf	right-down
exec nam out.nam &	set udp0 [new Agent/UDP]
}	\$udp0 set class_1
set n0 [\$ns node]	\$ns attach-agent \$n0 \$udp0
set n1 [\$ns node]	set cbr0 [new
set n2 [\$ns node]	Application/Traffic/CBR]
set n3 [\$ns node]	\$cbr0 set packetsize_ 500
set n4 [\$ns node]	\$cbr0 set interval_ 0.005
\$ns duplex-link \$n0 \$n1 1Mb 10ms	\$cbr0 attach-agent \$udp0
DropTail	set udp1 [new Agent/UDP]
\$ns duplex-link \$n0 \$n2 1Mb 10ms	<pre>\$udp1 set class_2</pre>
DropTail	\$ns attach-agent \$n1 \$udp1
\$ns duplex-link \$n0 \$n3 1Mb 10ms	set cbr1 [new
DropTail	Application/Traffic/CBR]
\$ns duplex-link \$n0 \$n4 1Mb 10ms	\$cbr1 set packetsize_ 500
DropTail	\$cbr1 set interval_ 0.005
\$ns duplex-link \$n1 \$n2 1Mb 10ms	\$cbr1 attach-agent \$udp1
DropTail	set null0 [new Agent/Null]
\$ns duplex-link \$n1 \$n3 1Mb 10ms	\$ns attach-agent \$n4 \$null0
DropTail	\$ns connect \$udp0 \$null0
\$ns duplex-link \$n1 \$n4 1Mb 10ms	\$ns connect \$udp1 \$null0
DropTail	\$ns at 0.5 "\$cbr0 start"
\$ns duplex-link \$n2 \$n3 1Mb 10ms	\$ns at 1.0 "\$cbr1 start"
DropTail	\$ns at 1.5 "\$cbr0 stop"
\$ns duplex-link \$n2 \$n4 1Mb 10ms	\$ns at 2.0 "\$cbr1 stop"
DropTail	\$ns at 2.5 "finish"
\$ns duplex-link \$n3 \$n4 1Mb 10ms	\$ns run
DropTail	