Week 4(CRC)

```
#include <stdio.h>
#include <string.h>
void main() {
        int i,j,keylen,msglen,l=0;
        char
input[100],key[30],temp[30],quot
[100],rem[30],key1[30],fdata[60];
        printf("Enter Data: ");
        gets(input);
        printf("Enter Divisor: ");
        gets(key);
        keylen=strlen(key);
        msglen=strlen(input);
        strcpy(key1,key);
        for (i=0;i<keylen-1;i++) {

        input[msglen+i]='0';
        }
        for (i=0;i<keylen;i++)
         temp[i]=input[i];
        for (i=0;i<msglen;i++) {
                quot[i]=temp[0];
                if(quot[i]=='0')
         for (j=0;j<keylen;j++)
                 key[j]='0'; else
         for (j=0;j<keylen;j++)
                 key[j]=key1[j];
        for (j=keylen-1;j>0;j--) {
        if(temp[j]==key[j])
                 rem[j-1]='0';
          else
                 rem[j-1]='1';
                }
rem[keylen1]=input[i+keylen];
                strcpy(temp,rem);
        }
        strcpy(rem,temp);
        printf("\nQuotient is ");
        for (i=0;i<msglen;i++)
         printf("%c",quot[i]);
        printf("\nRemainder is ");
        for (i=0;i<keylen-1;i++)
         printf("%c",rem[i]);
        for (i=0;i<msglen;i++)
 {
  fdata[l]=input[i];
  l++;
 }
        for (i=0;i<keylen-1;i++)
 {
  fdata[l]=rem[i];
  l++;
 }
  printf("\nFinal data is: ");
  for(i=0;i<l;i++)
         printf("%c",fdata[i])
}
```

Week 4 (Hamming)

```
#include <stdio.h>
#include <math.h>
int input[32];
int code[32];
int ham_calc(int,int);
void main()
{      int n,i,p_n = 0,c_l,j,k;
          printf("Please enter the
length of the Data Word: ");
          scanf("%d",&n);
          printf("Please enter the
Data Word:\n");
          for(i=0;i<n;i++)
{ scanf("%d",&input[i]); }

i=0;
          while(n>(int)pow(2,i)-(i+1))
          {    p_n++;
                    i++;}
          c_l = p_n + n;
          j=k=0;
          for(i=0;i<c_l;i++)
          {
          if(i==((int)pow(2,k)-1))
                    {   code[i]=0;
                              k++;
                    }
                    else
                    {  code[i]=input[j];
                              j++;
                    } }
          for(i=0;i<p_n;i++)
          {
          int position = (int)pow(2,i);
int value = ham_calc(position,c_l);
          code[position-1]=value;
          }
          printf("\nThe calculated
Code Word is: ");
          for(i=0;i<c_l;i++)
          printf("%d",code[i]);
          printf("\n");
          printf("Please enter the
received Code Word:\n");
          for(i=0;i<c_l;i++)

          scanf("%d",&code[i]);

          int error_pos = 0;
          for(i=0;i<p_n;i++)
          {
          int position = (int)pow(2,i);
int value = ham_calc(position,c_l);
                    if(value != 0)
          error_pos+=position;
          }
          if(error_pos == 1)
          printf("The received Code
Word is correct.\n");
          else
               printf("Error at bit
position: %d\n",error_pos);
}
int ham_calc(int position,int c_l)
{
          int count=0,i,j;
          i=position-1;
          while(i<c_l)
          { for(j=i;j<i+position;j++)
                    {
                    if(code[j] == 1)
                              count++;
                    }
                    i=i+2*position;
          }
          if(count%2 == 0)
                    return 0;
          else
                    return 1;  }
```

#PROGRAM: CSMA/CA

```
set ns [new Simulator]
$ns color 1 Blue
$ns color 2 Red
set file1 [open out.tr w]
set winfile [open WinFile w]
$ns trace-all $file1
set file2 [open out.nam w]
$ns namtrace-all $file2
proc finish {} {
global ns file1 file2
$ns flush-trace
close $file1
close $file2
exec nam out.nam &
exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
$n1 color red
$n1 shape box
$ns duplex-link $n0 $n2 2Mb 10ms
DropTail
$ns duplex-link $n1 $n2 2Mb 10ms
DropTail
$ns simplex-link $n2 $n3 0.3Mb
100ms DropTail
$ns simplex-link $n3 $n2 0.3Mb
100ms DropTail
set lan [$ns newLan "$n3 $n4 $n5"
0.5Mb 40ms LL Queue/DropTail
MAC/Csma/Ca Channel]
Setup a TCP connection
set tcp [new Agent/TCP/Newreno]
$ns attach-agent $n0 $tcp
set sink [new
Agent/TCPSink/DelAck]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
$tcp set window_ 8000
$tcp set packetSize_ 552
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n5 $null
$ns connect $udp $null
$udp set fid_ 2
set cbr [new
Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 0.01mb
$cbr set random_ false
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 124.0 "$ftp stop"
proc plotWindow {tcpSource file} {
global ns
set time 0.1
set now [$ns now]
set cwnd [$tcpSource set cwnd_]
set wnd [$tcpSource set window_]
puts $file "$now $cwnd"
$ns at [expr $now+$time]
"plotWindow $tcpSource $file" }
$ns at 0.1 "plotWindow $tcp
$winfile"
$ns at 5 "$ns trace-annotate
\"packet drop\""
$ns at 125.0 "finish"      $ns run
```

#PROGRAM: CSMA/CD

```
set ns [new Simulator]
$ns color 1 Blue
$ns color 2 Red
set file1 [open out.tr w]
set winfile [open WinFile w]
$ns trace-all $file1
set file2 [open out.nam w]
$ns namtrace-all $file2
proc finish {} {
global ns file1 file2
$ns flush-trace
close $file1
close $file2
exec nam out.nam &
exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
$n1 color red
$ns duplex-link $n0 $n2 2Mb 10ms
DropTail
$ns duplex-link $n1 $n2 2Mb 10ms
DropTail
$ns simplex-link $n2 $n3 0.3Mb
100ms DropTail
$ns simplex-link $n3 $n2 0.3Mb
100ms DropTail
set lan [$ns newLan "$n3 $n4 $n5"
0.5Mb 40ms LL Queue/DropTail
MAC/Csma/Ca Channel]
set tcp [new Agent/TCP/Newreno]
$ns attach-agent $n0 $tcp
set sink [new
Agent/TCPSink/DelAck]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
$tcp set window_ 8000
$tcp set packetSize_ 552
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n5 $null
$ns connect $udp $null
$udp set fid_ 2
set cbr [new
Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 0.01mb
$cbr set random_ false
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 124.0 "$ftp stop"
$ns at 124.5 "$cbr stop"
proc plotWindow {tcpSource file} {
global ns
set time 0.1
set now [$ns now]
set cwnd [$tcpSource set cwnd_]
set wnd [$tcpSource set window_]
puts $file "$now $cwnd"
$ns at [expr $now+$time]
"plotWindow $tcpSource $file" }
$ns at 0.1 "plotWindow $tcp
$winfile"
$ns at 5 "$ns trace-annotate
\"packet drop\""
$ns at 125.0 "finish"
$ns run
```

#stop and wait

```
include<stdio.h>
        int timer=0,wait_for_ack=-
1,frameQ=0,cansend=1,t=0;
         main()
        {
              int i,j,k;
              int frame[5];
              //clrscr();
              printf("enter the time
when data frame will be ready\n");
              for(j=0;j<3;j++)
              {
                    sender( i, &frame);
                    recv(i);
              }
        }
                    {
                          wait_for_ack++;
                      if(wait_for_ack==3)
                      {
                    }
                    if(i==frame[t])
                    {
                          frameQ++;
                          t++;
                    }
                    if(frameQ==0)
                          printf("NO
FRAME TO SEND at time=%d \n",i);
                    if(frameQ>0 &&
cansend==1)
                    {
                          printf("FRAME
SEND AT TIME=%d\n",i);
                          cansend=-1;
                          frameQ--;
                          timer++;
                          printf("timer in
sender=%d\n",timer);    }
if(frameQ>0 && cansend==-1)
        printf("FRAME IN Q FOR
TRANSMISSION AT TIME=%d\n",i);
if(frameQ>0)
             t++;
printf("frameQ=%d\n",frameQ);
printf("i=%d     t=%d\n",i,t);
printf("value in
frame=%d\n",frame[t]);
 //      return 0;
 }
recv(int i )
{   printf("timer recvr=%d\n",timer);
                    if(timer>0)
                    {
                          timer++;
                    }
                          if(timer==3)
                          {
                                printf("
                    FRAME ARRIVED AT
TIME=%d\n",i);

wait_for_ack=0;
                                timer=0;
                          }
                          else
                                printf("
                    WAITING FOR FRAME
AT TIME %d\n",i);
          //                    return 0;

                }
```

SAMPLE TOPOLOGY

```
set ns [new Simulator]
set nf [open out.nam w]
$ns namtrace-all $nf
proc finish {} {
        global ns nf
        $ns flush-trace
        close $nf
        exec nam out.nam &
        exit 0
}
set n0 [$ns node]
set n1 [$ns node]
$ns duplex-link $n0  $n1 1Mb 10ms
DropTail
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 $n1 $null0
$ns connect $udp0 $null0
$ns at 0.5 "$cbr0 start"
$ns at 2.5 "$cbr0 stop"
$ns at 3.0 "finish"
$ns run
```

BUS TOPOLOGY

```
set ns [new Simulator]
$ns color 1 Blue
set nf [open out.nam w]
$ns namtrace-all $nf
proc finish {} {
        global ns nf
        $ns flush-trace
        close $nf
        exec nam out.nam &
        exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$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-op $n0 $n1 orient
right
$ns duplex-link-op $n1 $n2 orient
right
$ns duplex-link-op $n2 $n3 orient
right
set udp0 [new Agent/UDP]
$udp0 set class_ 1
$ns attach-agent $n0 $udp0
set cbr0 [new
Application/Traffic/CBR]
$cbr0 set package_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
set null0 [new Agent/Null]
$ns attach-agent $n2 $null0
$ns connect $udp0 $null0
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
$ns at 5.0 "finish"
$ns run
```

RING TOPOLOGY

```
set ns [new Simulator]
$ns color 1 Blue
$ns color 2 Red
$ns color 3 Green
$ns color 4 Black
set nf [open out.nam w]
$ns namtrace-all $nf
proc finish {} {
        global ns nf
        $ns flush-trace
        close $nf
        exec nam out.nam &
        exit 0     }
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
$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
$ns duplex-link $n4 $n5 1Mb 10ms
DropTail
$ns duplex-link $n5 $n0 1Mb 10ms
DropTail
$ns duplex-link-op $n0 $n1 orient
right-up
$ns duplex-link-op $n1 $n2 orient
right
$ns duplex-link-op $n2 $n3 orient
right-down
$ns duplex-link-op $n3 $n4 orient
left-down
$ns duplex-link-op $n4 $n5 orient
left
$ns duplex-link-op $n5 $n0 orient
left-up
$ns duplex-link-op $n0 $n1
queuePos 0.0
set udp0 [new Agent/UDP]
$udp0 set class_ 1
$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 udp1 [new Agent/UDP]
$udp1 set class_ 2
$ns attach-agent $n1 $udp1
set cbr1 [new
Application/Traffic/CBR]
$cbr1 set packetsize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1
set udp2 [new Agent/UDP]
$udp2 set class_ 3
$ns attach-agent $n2 $udp2
set cbr2 [new
Application/Traffic/CBR]
$cbr2 set packetsize_ 500
$cbr2 set interval_ 0.005
$cbr2 attach-agent $udp2
set udp3 [new Agent/UDP]
$udp3 set class_ 4
$ns attach-agent $n3 $udp3
set cbr3 [new
Application/Traffic/CBR]
$cbr3 set packetsize_ 500
$cbr3 set interval_ 0.005
$cbr3 attach-agent $udp3
set udp4 [new Agent/UDP]
$udp4 set class_ 5
```

```
$ns attach-agent $n4 $udp4
set cbr4 [new
Application/Traffic/CBR]
$cbr4 set packetsize_ 500
$cbr4 set interval_ 0.005
$cbr4 attach-agent $udp4
set null0 [new Agent/Null]
$ns attach-agent $n5 $null0
$ns connect $udp0 $null0
$ns connect $udp1 $null0
$ns connect $udp2 $null0
$ns connect $udp3 $null0
$ns connect $udp4 $null0
$ns at 0.1 "$cbr0 start"
$ns at 0.5 "$cbr1 start"
$ns at 1.0 "$cbr2 start"
$ns at 1.5 "$cbr3 start"
$ns at 2.0 "$cbr4 start"
$ns at 3.5 "$cbr4 stop"
$ns at 4.0 "$cbr3 stop"
$ns at 4.5 "$cbr2 stop"
$ns at 5.0 "$cbr1 stop"
$ns at 5.5 "$cbr0 stop"
$ns at 6.0 "finish"
$ns run
```

STAR TOPOLOGY

```
set ns [new Simulator]
$ns color 1 Blue
$ns color 2 Red
set nf [open out.nam w]
$ns namtrace-all $nf
proc finish {} {
        global ns nf
        $ns flush-trace
        close $nf
        exec nam out.nam &
        exit 0    }
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
$ns duplex-link $n0 $n2 1Mb 10ms
DropTail
$ns duplex-link $n1 $n2 1Mb 10ms
DropTail
$ns duplex-link $n3 $n2 1Mb 10ms
DropTail
$ns queue-limit $n2 $n3 10
$ns duplex-link-op $n0 $n2 orient
right-up
$ns duplex-link-op $n1 $n2 orient
right-down
$ns duplex-link-op $n2 $n3 orient
right
$ns duplex-link-op $n2 $n3
queuePos 0.5
set udp0 [new Agent/UDP]
$udp0 set class_ 1
$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 udp1 [new Agent/UDP]
$udp1 set class_ 2
$ns attach-agent $n1 $udp1
set cbr1 [new
Application/Traffic/CBR]
$cbr1 set packetsize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1
set null0 [new Agent/Null]
$ns attach-agent $n3 $null0
$ns connect $udp0 $null0
$ns connect $udp1 $null0
$ns at 0.5 "$cbr0 start"
$ns at 1.0 "$cbr1 start"
$ns at 4.0 "$cbr1 stop"
$ns at 4.5 "$cbr0 stop"
```

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

MESH TOPOLOGY

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