Computer Network Practical Exam

SET-1

NAME : AKSHAT KUSHWAH

EXAM ROLL NO. : 2002057004

COLLEG ROLL NO. : 20201403

QUESTION1. Simulate Cyclic Redundancy Check (CRC) error detection algorithm for noisy channel.

CODE:

```
//Question 1.Simulate Cyclic Redundancy Check (CRC) error detection algorithm for
noisy channel.

// Description:
//   Cyclic Redundancy Check (CRC) is a error detection technique used in data
link layer
//   In CRC algorithm :
//     Sender's side:
//        If l is the length of the divisor (in Binary)
//        Then l-1 '0' bits are appended to the original messsage
//        Then the appended message is divided by the divisor
//        Then resulting (l-1) remainder is appended instesd of '0'
```

```
//        Message sent

//     Reciever's side:
//        revieved message is divided by the divisor
//        If remainder : all '0s'     -> (No Error) "Message Acccepted"
//        If remainder : not all '0s' ->  (Error)   "Message Rejected"



#include<iostream>
using namespace std;
int gen,msg;
int lenght;
void rev(int num[],int len)
{
     int temp;
     for(int i1=0,i2=len-1;i1<len/2;i1++,i2--)
     {
             temp=num[i2];
             num[i2]=num[i1];
             num[i1]=temp;
     }
     return;
}
void dec_to_bin(int dec,int bin_ary[])
{
       int tmp,i=0;
       tmp=dec;
       while(tmp!=0)
       {
                    bin_ary[i]=tmp%2;
                    tmp=tmp/2;
                    i++;
       }
       rev(bin_ary,i);
       lenght=i;
       for(int a=0;a<i;a++)
               cout<<bin_ary[a];
       cout<<endl;
       return;
}
 int sub_xor_bin(int divt[],int divr[],int len_div,int len)
 {
      int j=0,k;
```

```
      while(divt[j]!=1)
                        j++;
      if((len-j)<len_div)
                         return 0;
      for(int i=j,k=0;i<(len_div+j);i++,k++)
      {
              if(divt[i]==divr[k])
                                  divt[i]=0;
              else
                  divt[i]=1;
      }
      return 1;
 }

 void app_end(int num[],int app,int &norm)
 {
      lenght=(norm+app)-1;
      for(int i=norm-1;i<lenght;i++)
              num[i]=0;
      cout<<"The number after appending is:\n";
      for(int i=0;i<lenght;i++)
              cout<<num[i]<<" ";
      cout<<endl;

      norm=lenght;
 }


 void division(int msg[],int &len_msg,int gn[],int len_gn)
 {
      int flag;
      app_end(msg,len_gn,len_msg);
      for(int i=0;i<len_msg-2;i++)
      {
              flag=sub_xor_bin(msg,gn,len_gn,len_msg);
              if(flag==0)
              break;
      }
      cout<<"The msg now after dividing:\n";
      for(int i=0;i<len_msg;i++)
              cout<<msg[i]<<" ";
      cout<<endl;
 }
int main()
{
```

```
      int
msg_pass[20],msg_temp[20],chc,i=0,lenght_msg,lenght_gen,rem[6],ori_msg_len;
      cout<<"Enter the message to be passed(in decimal):\n";
      cin>>msg;
   cout<<"The message in binary code:\n";
   dec_to_bin(msg,msg_pass);
      lenght_msg=lenght;
      cout<<"Enter the generator number i.e, the divisor:(in decimal)\n";
      cin>>gen;
      cout<<"the binary value of generator is:\n";
      int genr[6];
      dec_to_bin(gen,genr);
      lenght_gen=lenght;
      for(int j=0;j<lenght_msg;j++)
              msg_temp[j]=msg_pass[j];
      ori_msg_len=lenght_msg;
      division(msg_pass,lenght_msg,genr,lenght_gen);
      i=0;
      while(msg_pass[i]!=1)
      {
                           i++;
                           if(i>=lenght_msg)
                           break;
      }
      int r=ori_msg_len;
      if(i<lenght_msg)
      {
                     cout<<"The crc remainder is:\n";
                     for(int j=i;j<lenght_msg;j++)
                     {
                             cout<<msg_pass[j]<<" ";
                             msg_temp[r]=msg_pass[j];
                             r++;
                     }
                     cout<<endl;
      }
      cout<<"The msg to be passed:\n";
      for(i=0;i<r;i++)
                      cout<<msg_temp[i]<<" ";
      cout<<endl;
      return 0;
}
```

OUTPUT:

```
Enter the message to be passed(in decimal):
1234
The message in binary code:
10011010010
Enter the generator number i.e, the divisor:(in decimal)
5
the binary value of generator is:
101
The number after appending is:
1 0 0 1 1 0 1 0 0 1 0 0 0
The msg now after dividing:
0 0 0 0 0 0 0 0 0 0 0 0 1
The crc remainder is:
1
The msg to be passed:
1 0 0 1 1 0 1 0 0 1 0 1
PS F:\Acadmics\sem 3\Co. Net\Codes>
```

QUESTION2. Simulate and implement selective repeat sliding window protocol.

CODE:

```
// Question 2.Simulate and implement selective repeat sliding window protocol.

// Description:
//      It is the protocol for noice channel.
//      In this multiple frames can be send at a time.
//      Number of frames is depends of the window size.
//      In this each frame is assigned with a sequence no.

// Algorithm:
//      In this all frames are sent without an acknowledgement which lie in
window
//      If the ACK of last frame is recieved ,then window move one step forward
```

```
//      And new frame added in the window will sent
//      This cycle repeats Until - There is no more frame (Window starts
shrinking)
//                               - ACk lost (In this case on respective frame is
resent)



#include<iostream>

#include<conio.h>
#include<stdlib.h>
#include<time.h>
#include<math.h>
using namespace std;



#define TOT_FRAMES 500

#define FRAMES_SEND 10



class sel_repeat

{

private:

int fr_send_at_instance;

int arr[TOT_FRAMES];

int send[FRAMES_SEND];

int rcvd[FRAMES_SEND];

char rcvd_ack[FRAMES_SEND];

int sw;

int rw;       //tells expected frame

public:
```

```
void input();

void sender(int);

void receiver(int);

};

void sel_repeat::input()

{

int n;     //no. of bits for the frame

int m;    //no. of frames from n bits
int i;
cout<<"Enter the no. of bits for the sequence no. : ";

cin>>n;

m=pow(2,n);

int t=0;

fr_send_at_instance=(m/2);

for(i=0;i<TOT_FRAMES;i++)

{

arr[i]=t;

t=(t+1)%m;

}

for(i=0;i<fr_send_at_instance;i++)

{

send[i]=arr[i];

rcvd[i]=arr[i];
```

```
rcvd_ack[i]='n';

}

rw=sw=fr_send_at_instance;

sender(m);

}

void sel_repeat::sender(int m)

{

for(int i=0;i<fr_send_at_instance;i++)

{

if(rcvd_ack[i]=='n')

cout<<"SENDER : Frame "<<send[i]<<" is sent\n";

}

receiver(m);

cout<<endl;
}

void sel_repeat::receiver(int m)

{

time_t t;

int f;
int j;
int f1;

int a1;

char ch;

srand((unsigned)time(&t));
```

```
for(int i=0;i<fr_send_at_instance;i++)

{

if(rcvd_ack[i]=='n')

{

f=rand()%10;

//if f=5 frame is discarded for some reason

//else frame is correctly recieved

if(f!=5)

{

for(int j=0;j<fr_send_at_instance;j++)

if(rcvd[j]==send[i])

{

cout<<"\nReciever -> Frame : "<<rcvd[j]<<"\t(Recieved correctly)\n";

rcvd[j]=arr[rw];

rw=(rw+1)%m;

break;

}
int j;
if(j==fr_send_at_instance)

cout<<"\nReciever ->Duplicate Frame : "<<send[i]<<"\t(Discarded)\n";

a1=rand()%5;

//if a1==3 then ack is lost

//else recieved

if(a1==3)
```

```
{
cout<<"(acknowledgement -> "<<send[i]<<" lost)\n";
cout<<"(sender timeouts-->Resend the frame)\n";
rcvd_ack[i]='n';
}
else
{
cout<<"(acknowledgement -> "<<send[i]<<" recieved)\n";
rcvd_ack[i]='p';
}
}
else
{int ld=rand()%2;
//if =0 then frame damaged
//else frame lost
if(ld==0)
{
cout<<"RECEIVER : Frame "<<send[i]<<" is damaged\n";
cout<<"RECEIVER : Negative Acknowledgement "<<send[i]<<" sent\n";
}
else
{
```

```
cout<<"RECEIVER : Frame "<<send[i]<<" is lost\n";

cout<<"(SENDER TIMEOUTS-->RESEND THE FRAME)\n";

}

rcvd_ack[i]='n';

}

}

}

for(int j=0;j<fr_send_at_instance;j++)

{

if(rcvd_ack[j]=='n')

break;

}

int i=0;

for(int k=j;k<fr_send_at_instance;k++)

{

send[i]=send[k];

if(rcvd_ack[k]=='n')

rcvd_ack[i]='n';

else

rcvd_ack[i]='p';

i++;

}

if(i!=fr_send_at_instance)
```

```
{

for(int k=i;k<fr_send_at_instance;k++)

{

send[k]=arr[sw];

sw=(sw+1)%m;

rcvd_ack[k]='n';

}

}
cout<<"\ny-> YES\n";
cout<<"n-> NO\n";
cout<<"Want to continue : ";

cin>>ch;

cout<<"\n";

if(ch=='y')

sender(m);

else
cout<<"\t EXIT.............\n";
exit(0);

}

int main()

{


sel_repeat sr;

sr.input();


}
```

OUTPUT:

```
Enter the no. of bits for the sequence no. : 4
SENDER : Frame 0 is sent
SENDER : Frame 1 is sent
SENDER : Frame 2 is sent
SENDER : Frame 3 is sent
SENDER : Frame 4 is sent
SENDER : Frame 5 is sent
SENDER : Frame 6 is sent
SENDER : Frame 7 is sent

Reciever -> Frame : 0   (Recieved correctly)
(acknowledgement -> 0 recieved)

Reciever -> Frame : 1   (Recieved correctly)
(acknowledgement -> 1 lost)
(sender timeouts-->Resend the frame)

Reciever -> Frame : 2   (Recieved correctly)
(acknowledgement -> 2 recieved)

Reciever -> Frame : 3   (Recieved correctly)
(acknowledgement -> 3 recieved)

Reciever -> Frame : 4   (Recieved correctly)
(acknowledgement -> 4 lost)
(sender timeouts-->Resend the frame)

Reciever -> Frame : 5   (Recieved correctly)
(acknowledgement -> 5 recieved)

Reciever -> Frame : 6   (Recieved correctly)
(acknowledgement -> 6 lost)
(sender timeouts-->Resend the frame)

Reciever -> Frame : 7   (Recieved correctly)
(acknowledgement -> 7 lost)
(sender timeouts-->Resend the frame)

y-> YES
n-> NO
Want to continue : y
```

```
SENDER : Frame 8 is sent
SENDER : Frame 9 is sent
SENDER : Frame 10 is sent
SENDER : Frame 11 is sent
SENDER : Frame 12 is sent
SENDER : Frame 13 is sent
SENDER : Frame 14 is sent
SENDER : Frame 15 is sent

Reciever -> Frame : 8   (Recieved correctly)
(acknowledgement -> 8 recieved)

Reciever -> Frame : 9   (Recieved correctly)
(acknowledgement -> 9 lost)
(sender timeouts-->Resend the frame)

Reciever -> Frame : 10  (Recieved correctly)
(acknowledgement -> 10 recieved)

Reciever -> Frame : 11  (Recieved correctly)
(acknowledgement -> 11 lost)
(sender timeouts-->Resend the frame)

Reciever -> Frame : 12  (Recieved correctly)
(acknowledgement -> 12 recieved)

Reciever -> Frame : 13  (Recieved correctly)
(acknowledgement -> 13 recieved)

Reciever -> Frame : 14  (Recieved correctly)
(acknowledgement -> 14 lost)
(sender timeouts-->Resend the frame)

Reciever -> Frame : 15  (Recieved correctly)
(acknowledgement -> 15 lost)
(sender timeouts-->Resend the frame)

y-> YES
n-> NO
Want to continue : n

        EXIT.............
```