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++)

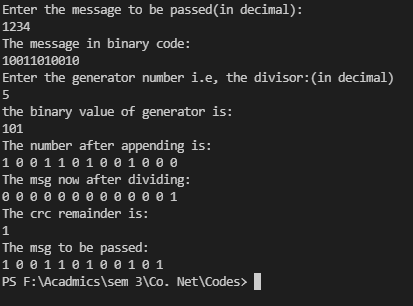
                       cout<<msg\_temp[i]<<" ";

      cout<<endl;

      return 0;

}

**OUTPUT:**

****

**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 al==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()

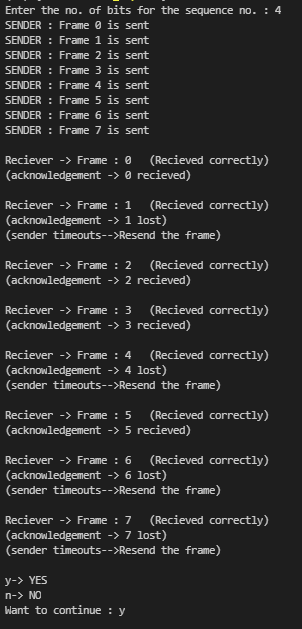
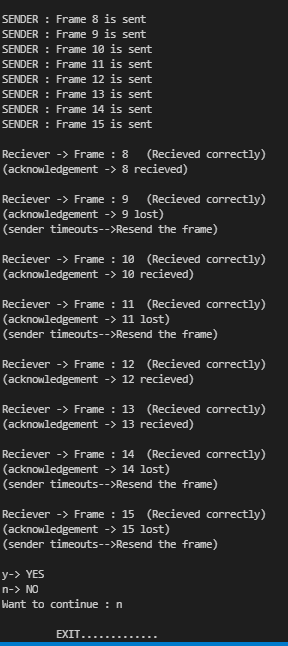
{

sel\_repeat sr;

sr.input();

}

**OUTPUT:**

****