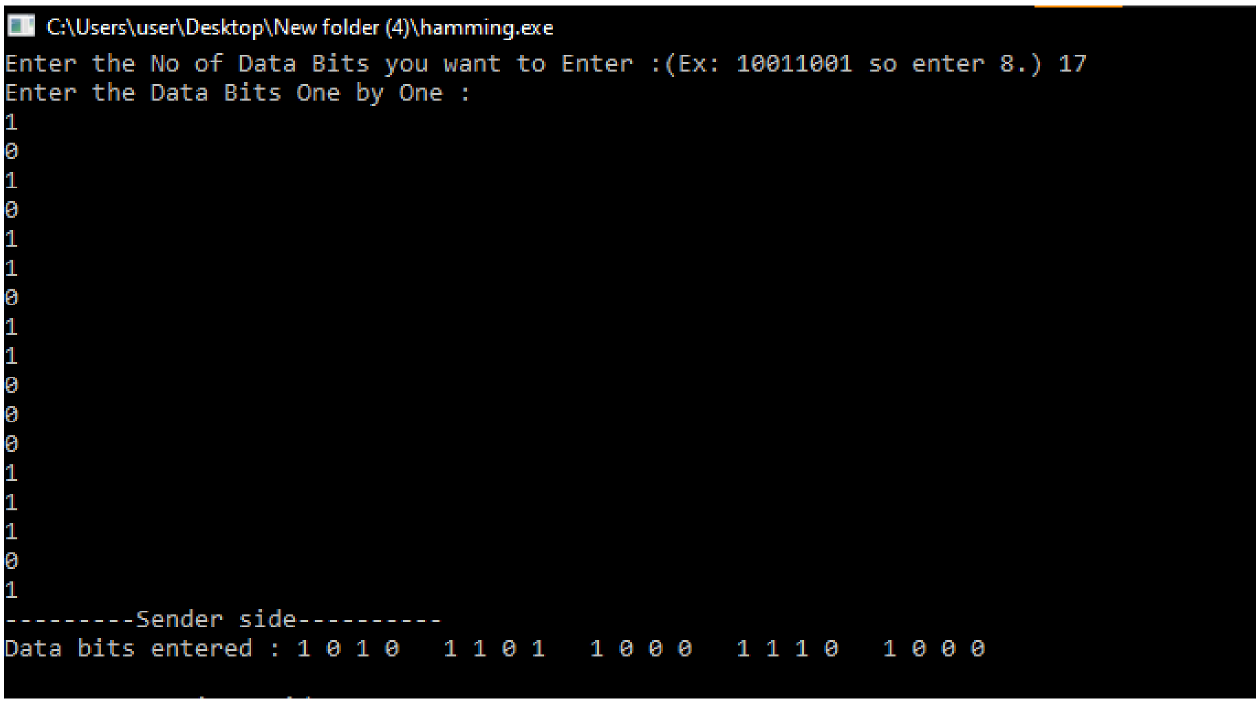
Cyclic Redundancy Check (CRC)

Code:

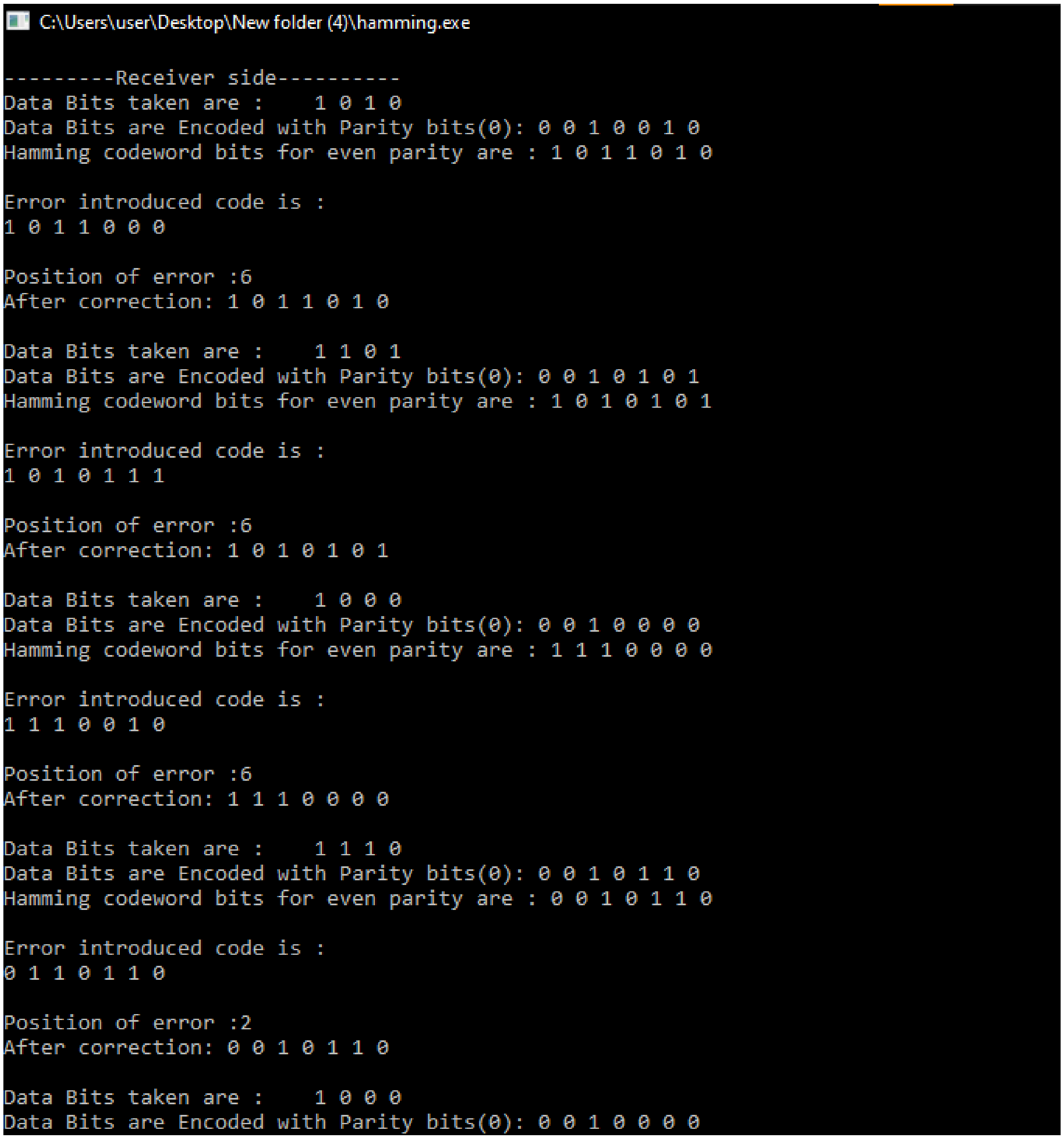
```
//cyclic redundancy check
#include <bits/stdc++.h>
#include <conio.h>
using namespace std;
int n,k;

string XOR(string data,string div,int i){
for(int k=i;k<i+div.length();k++)
{if(data[k]==div[k-i])
  data[k]='0';
 else
  data[k]='1';}
  //cout<<data<<endl;
  return data;
  }

//encoding
string encode(string data,string div){
  n= data.length(),k=div.length();
 for(int i=1;i<k;i++)
   data+='0';
 //cout << "Augmented Data is  "<<data<<endl<<endl;
 string rem=data;
 for(int i=0;i<n;i++)
  {if(rem[i]=='0')
    continue;
   else
    rem= XOR(rem,div,i);}
 cout<<endl;

 string code=XOR(data,rem.substr(n,k-1),n);



 return code;
 }


//decoding
int decode(string code,string div){
 n= code.length();
 k= div.length();
 for(int i=0;i<n-k+1;i++)
 {if(code[i]=='0')
    continue;
  else
   code= XOR(code,div,i) ;
 }
//cout<<"So,Final remainder is "<<code<<endl;
int zeroes=0,err=0;
for(int i=n-k;i<n;i++)
{if(code[i]=='0')
 zeroes++;}
 if(zeroes==k)
  cout<<"NO ERROR detected using CRC"<<endl;
 else
  {cout<<"ERROR detected using CRC"<<endl<<"DISCARD"<<endl;
   err=1;}
return err;

}

string error(string data,float p){

for(int i=0;i<data.length();i++)
{float r = ((float) rand() / (RAND_MAX));
if(r<p)
{if(data[i]=='0')
 data[i]='1';
 else
 data[i]='0';}
}
return data;
}
```

```
int main() {
//encoding
 string data,div="100000111",code;
 cout<<"Enter Data Stream"<<endl;
 cin >> data;

 cout<<"CRC-8 Divisor is "<<div<<endl;
   n= data.length(),k=div.length();


 float token=16.0,block=n/token;
 int a=n%(int)token;
 if(n%(int)token==0)
 {block-=0.5;
 a=token;}
 string arr[(int)ceil(block)];


 for(int i=1;i<=floor(block);i++)
 arr[i]=data.substr(a+token*(i-1),token);

 arr[0]=data.substr(0,a);
 int t=arr[0].length();
 for(int i=0;i<token-t;i++)
 arr[0]='0'+arr[0];

string codearr[(int)ceil(block)];

for(int i=0;i<=floor(block);i++)
{codearr[i]=encode(arr[i],div);
 cout<<token<<" bit Tokenized data "<<i+1 << " is   : "<<arr[i]<<endl;
 cout<<"CodeWord "<<i+1<<" at sender site is : "<< codearr[i]<< endl;
  cout<<endl;}

int hops;
float p;
cout<<"Enter no of hops in binary symmetric channel : (1 0r 2) ";
cin>>hops;
cout<<endl<<"Enter crossover probability for binary symmetric channel :";
cin>>p;
cout<<endl;
string errarr[(int)ceil(block)];
for(int i=0;i<=floor(block);i++)
  errarr[i]=codearr[i];
for(int i=0;i<hops;i++)
{for(int i=0;i<=floor(block);i++)
  errarr[i]=error(errarr[i],p);


 }
int errno;
for(int i=0;i<=floor(block);i++)
{cout<<"Code Word send     "<<i <<" is " <<codearr[i]<<endl;
 cout<<"Code Word recieved "<<i <<" is "<<errarr[i]<<endl;
 errno+=decode(errarr[i],div);
 cout<<endl;}

if(errno>0)
cout<<"Message is Discarded"<<endl;
else
{cout<<"NO Error in recieved data & Extracted Data from Code Word is
"<<endl;
for(int i=0;i<=floor(block);i++)
 cout<<errarr[i].substr(0,token);
 cout<<endl;}

 cout<<data;
 cout<<" was our original data     ";

// code=encode(data,div);
 getch();

 }
```

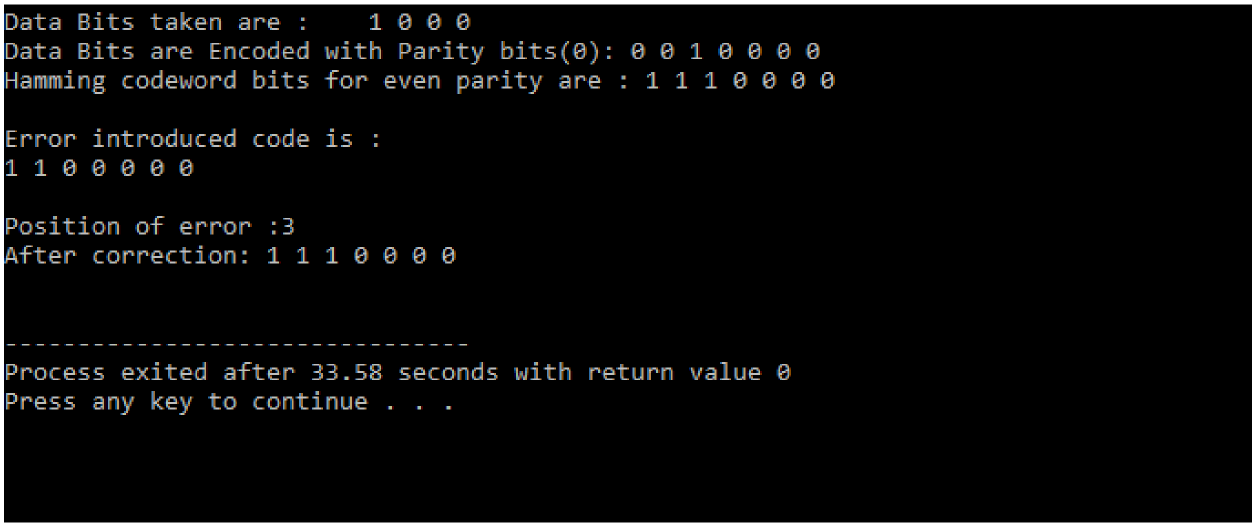
OUTPUT



```
C:\Users\user\Desktop\New folder (4)\hamming.exe
Enter the No of Data Bits you want to Enter :(Ex: 10011001 so enter 8.) 17
Enter the Data Bits One by One :
1
0
1
0
1
1
0
1
1
0
0
0
1
1
1
0
1
---------Sender side----------
Data bits entered : 1 0 1 0   1 1 0 1   1 0 0 0   1 1 1 0   1 0 0 0
```


C:\Users\user\Desktop\New folder (4)\hamming.exe

```
---------Receiver side----------
Data Bits taken are :    1 0 1 0
Data Bits are Encoded with Parity bits(0): 0 0 1 0 0 1 0
Hamming codeword bits for even parity are : 1 0 1 1 0 1 0

Error introduced code is :
1 0 1 1 0 0 0

Position of error :6
After correction: 1 0 1 1 0 1 0

Data Bits taken are :    1 1 0 1
Data Bits are Encoded with Parity bits(0): 0 0 1 0 1 0 1
Hamming codeword bits for even parity are : 1 0 1 0 1 0 1

Error introduced code is :
1 0 1 0 1 1 1

Position of error :6
After correction: 1 0 1 0 1 0 1

Data Bits taken are :    1 0 0 0
Data Bits are Encoded with Parity bits(0): 0 0 1 0 0 0 0
Hamming codeword bits for even parity are : 1 1 1 0 0 0 0

Error introduced code is :
1 1 1 0 0 1 0

Position of error :6
After correction: 1 1 1 0 0 0 0

Data Bits taken are :    1 1 1 0
Data Bits are Encoded with Parity bits(0): 0 0 1 0 1 1 0
Hamming codeword bits for even parity are : 0 0 1 0 1 1 0

Error introduced code is :
0 1 1 0 1 1 0

Position of error :2
After correction: 0 0 1 0 1 1 0

Data Bits taken are :    1 0 0 0
Data Bits are Encoded with Parity bits(0): 0 0 1 0 0 0 0
```



```
Data Bits taken are :    1 0 0 0
Data Bits are Encoded with Parity bits(0): 0 0 1 0 0 0 0
Hamming codeword bits for even parity are : 1 1 1 0 0 0 0

Error introduced code is :
1 1 0 0 0 0 0

Position of error :3
After correction: 1 1 1 0 0 0 0



--------------------------------
Process exited after 33.58 seconds with return value 0
Press any key to continue . . .
```

7-BIT HAMMING CODE

Code:

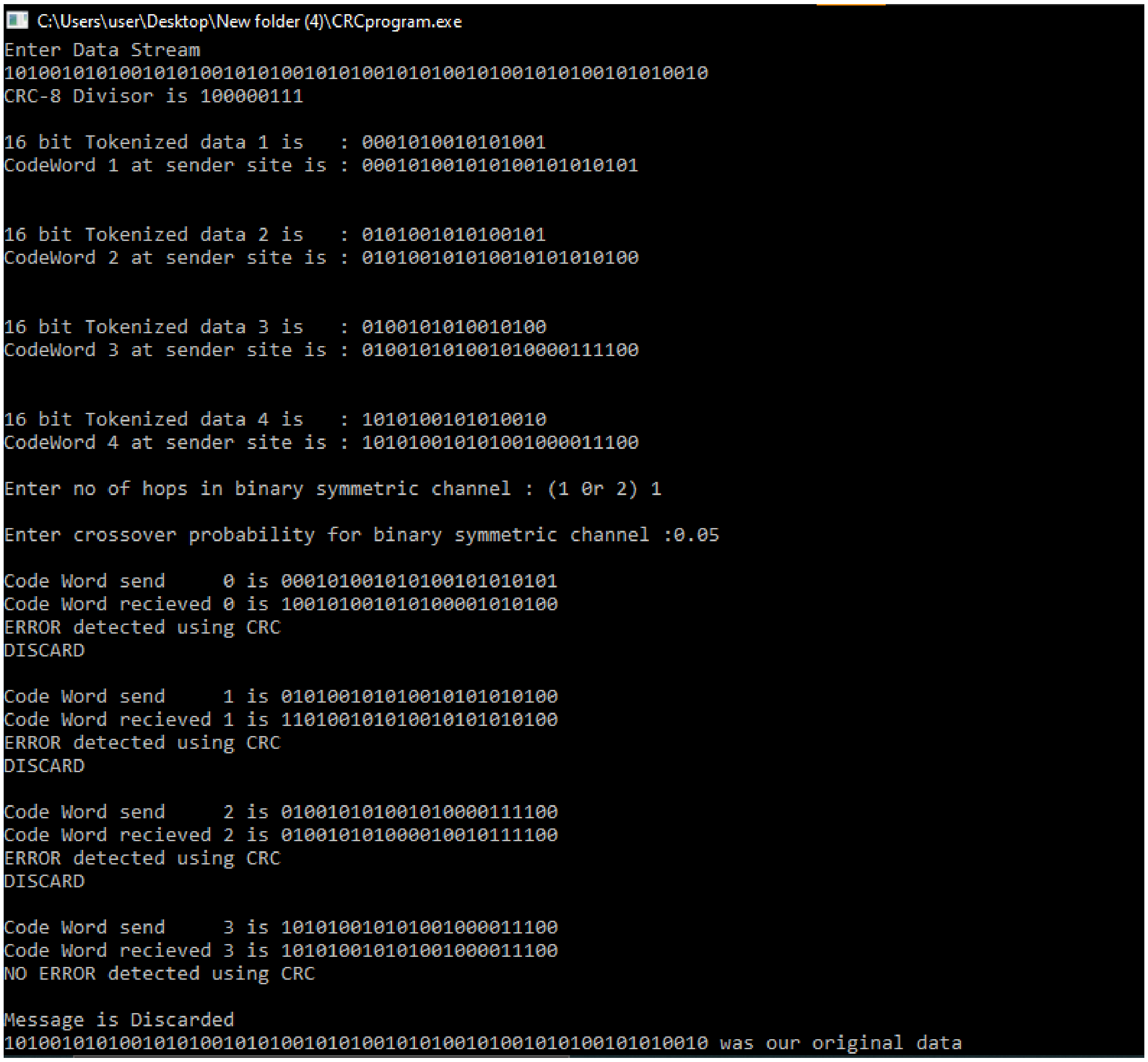
```
#include <iostream>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <bits/stdc++.h>
using namespace std;
void ham(int a, int r,int c[]);
int a, b, c[30], d, r = 0, d1,r1,rem,k,err[10]={0}; //Max bits here i kept is 30
int main ()
{
        srand(time(0));
    cout << "Enter the No of Data Bits you want to Enter :(Ex: 10011001 so enter
8.) ";
    cin >> a;
    cout << "Enter the Data Bits One by One :" << endl;
    for (int i = 1; i <= a; ++i)
        cin >> c[i];
        rem=4-(a%4);
        for(int i=1; i<=rem;++i)
        c[a+i]=0;
        cout<<"---------Sender side----------";
    cout << endl << "Data bits entered : ";
    for (int i = 1; i <= a+rem; ++i)
        {
                cout << c[i] << " ";
        if(i%4==0)
        cout<<"  ";
        }
    cout << endl;
    int fix=0;
    cout<<endl<<"---------Receiver side----------"<<endl;
    int tempham[5];
    for(int i=1;i<=((a+rem)/4);i++)
    {
         tempham[1]=c[1+fix];
         tempham[2]=c[2+fix];
         tempham[3]=c[3+fix];
```

```
          tempham[4]=c[4+fix];
          ham(4,3,tempham);
          fix=fix+4;
          }
}

    void ham(int a, int r,int c[])
    {
    int data[a + r], res[a+r];
    d = 0;
    d1 = 1;
    for (int i = 1; i <= a + r; ++i)
    {
      if ((i) == pow (2, d))
        {
        data[i] = 0;
        ++d;
        }
      else
        {
        data[i] = c[d1];
        ++d1;
        }
    }
    cout<<"Data Bits taken are : \t ";
    for(int i=1;i<=4;i++)
    cout<<c[i]<<" ";
    cout<<endl;
    cout << "Data Bits are Encoded with Parity bits(0): ";
    for (int i = 1; i <= a + r; ++i)
        cout << data[i] << " ";


    d1 = 0;
    int min, max = 0, parity, s, j;
    /*Parity Bit Calculation */
    for (int i = 1; i <= a + r; i = pow (2, d1))
    {
      ++d1;
      parity = 0;
      j = i;
      s = i;
```

```
  min = 1;
  max = i;
  for (j; j <= a + r;)
{
    for (s = j; max >= min && s <= a + r; ++min, ++s)
    {
    if (data[s] == 1)
        parity++;
    }
    j = s + i;
    min = 1;
}
  if (parity % 2 == 0) // Even Parity
{
data[i] = 0;
}
  else
{
data[i] = 1;
}
}
cout << endl << "Hamming codeword bits for even parity are : ";
for (int i = 1; i <= a + r; ++i)
    cout << data[i] << " ";
cout << endl << endl;
for (int i = 1; i <= a + r; ++i)
    res[i]=data[i];
k=(rand()%(a+r))+1;
      if(res[k]==0)
      res[k]=1;
      else
      res[k]=0;
      cout<<"Error introduced code is :"<<endl;
      for (int i = 1; i <= a + r; ++i)
    cout << res[i] << " ";
cout << endl;
d1 = 0;max=0;int ec=0;
//int min, max = 0, parity, s, j;
/*Parity Bit Calculation */
for (int i = 1; i <= a + r; i = pow (2, d1))
{
  ++d1;
  parity = 0;
  j = i;
  s = i;
  min = 1;
  max = i;
  for (j; j <= a + r;)
{
   for (s = j; max >= min && s <= a + r; ++min, ++s)
   {
   if (res[s] == 1)
       parity++;
   }
   j = s + i;
   min = 1;
}
  if (parity % 2 == 0) // Even Parity
{
err[ec]=0;
ec++;
}
  else
{
err[ec]=1;
ec++;
}
}
int flag = 1;
for(int i =r-1;i>=0;i--)
{
   if(err[i]==1)
   {
       flag =0;
           break;
   }
}
if(flag==0)
{
   int pos=0;
for(int i =r-1;i>=0;i--)
{
   if(err[i]==1)
       pos+=pow(2,i);
```

```
    }
        cout<<"\nPosition of error :"<<pos;
        res[pos]=!res[pos];
        cout<<"\nAfter correction: ";
        for(int i =1;i<=a+r;i++)
          cout<<res[i]<<" ";
          cout<<endl<<endl;
    }
    else
       cout<<"No Error detected. ";

}


//End
```

OUTPUT



C:\Users\user\Desktop\New folder (4)\CRCprogram.exe

```
Enter Data Stream
1010010101001010100101010010101001010100101001010100101010010
CRC-8 Divisor is 100000111

16 bit Tokenized data 1 is   : 0001010010101001
CodeWord 1 at sender site is : 000101001010100101010101


16 bit Tokenized data 2 is   : 0101001010100101
CodeWord 2 at sender site is : 010100101010010101010100


16 bit Tokenized data 3 is   : 0100101010010100
CodeWord 3 at sender site is : 010010101001010000111100


16 bit Tokenized data 4 is   : 1010100101010010
CodeWord 4 at sender site is : 101010010101001000011100

Enter no of hops in binary symmetric channel : (1 0r 2) 1

Enter crossover probability for binary symmetric channel :0.05

Code Word send     0 is 000101001010100101010101
Code Word recieved 0 is 100101001010100001010100
ERROR detected using CRC
DISCARD

Code Word send     1 is 010100101010010101010100
Code Word recieved 1 is 110100101010010101010100
ERROR detected using CRC
DISCARD

Code Word send     2 is 010010101001010000111100
Code Word recieved 2 is 010010101000010010111100
ERROR detected using CRC
DISCARD

Code Word send     3 is 101010010101001000011100
Code Word recieved 3 is 101010010101001000011100
NO ERROR detected using CRC

Message is Discarded
1010010101001010100101010010101001010100101001010100101010010 was our original data
```