Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Course Code: BCSE308P

Course Name: Computer Networks Lab

In Lab Assessment – 3

Name: Shivam Dave

Reg. No.: 21BCB0107

Problem Statement:

To implement error detection (CRC) and error correction techniques(Hamming code).

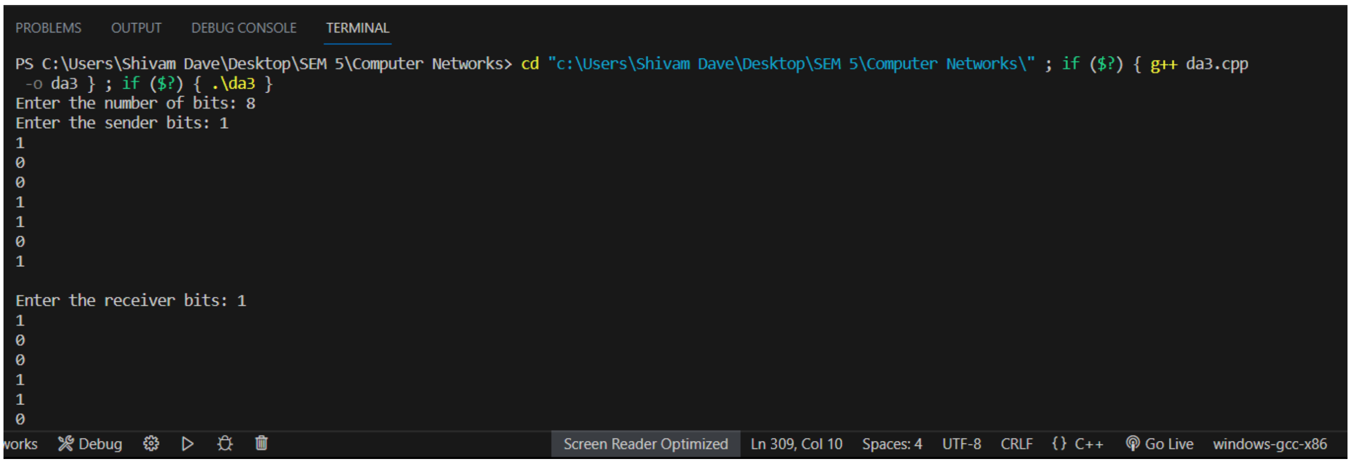
Code:

```
#include<iostream>
#include<vector>
#include<algorithm>
using namespace std;
void hamming(int sender[],int receiver[],int n)//hamming code
{
    int paritybitS1 = sender[0] xor sender[1] xor sender[3] xor sender[4] xor
sender[6];
    int paritybitS2 = sender[0] xor sender[2] xor sender[3] xor sender[5] xor
sender[6];
    int paritybitS3 = sender[1] xor sender[2] xor sender[3] xor sender[7];
    int paritybitS4 = sender[4] xor sender[5] xor sender[6] xor sender[7];

    int paritybitR1 = receiver[0] xor receiver[1] xor receiver[3] xor
receiver[4] xor receiver[6];
    int paritybitR2 = receiver[0] xor receiver[2] xor receiver[3] xor
receiver[5] xor receiver[6];
    int paritybitR3 = receiver[1] xor receiver[2] xor receiver[3] xor
receiver[7];
    int paritybitR4 = receiver[4] xor receiver[5] xor receiver[6] xor
receiver[7];

    int syndrome1 = paritybitR1 xor paritybitS1;
    int syndrome2 = paritybitR2 xor paritybitS2;
    int syndrome3 = paritybitR3 xor paritybitS3;
    int syndrome4 = paritybitR4 xor paritybitS4;

    int Syndrome = syndrome4*(8) + syndrome3*(4) + syndrome2*(2) +
syndrome1*(1);

    cout<<"Syndrome is: "<<Syndrome<<endl;

    int pos = -1;
    int parityPos = -1;
    switch (Syndrome){
    case 1:{
        pos = -2;
        parityPos = 1;
        break;
    }

    case 2:{
        pos = -2;
        parityPos = 2;
        break;
    }

    case 3:{
        pos = 1;
        // parityPos = 1;
        break;
    }

    case 4:{
        pos = -1;
        parityPos = 3;
        break;
    }

    case 5:{
        pos = 2;
        // parityPos = 1;
        break;
    }

    case 6:{
        pos = 3;
        // parityPos = 2;
        break;
    }

    case 7:{
        pos = 4;
        // parityPos = 1;
        break;
    }

    case 8:{
        pos = -1;
        parityPos = 4;
        break;
    }

    case 9:{
        pos = 5;
        // parityPos = 1;
        break;
    }

    case 10:{
        pos = 6;
        // parityPos = 2;
        break;
    }

    case 11:{
        pos = 7;
        // parityPos = 1;
        break;
    }

    case 12:{
        pos = 8;
        // parityPos = 2;
        break;
    }

    default:{
        pos = -1;
        parityPos = -1;
        break;
    }
    }

    if(pos == -1 && parityPos == -1){
        cout<<"No Error"<<endl;
    }
    if(parityPos != -1){
        cout<<"Parity position is wrong at position: "<<parityPos<<endl;
    }else{
        cout<<endl;
        cout<<"The sender bits were:   ";
        for(int i = n-1; i>=0; i--){
            cout<<sender[i];
        }
        cout<<endl;
        cout<<"The receiver bits were: ";
        for(int i = n-1; i>=0; i--){
            cout<<receiver[i];
        }
        cout<<endl;
        cout<<"Data bit wrong position: "<<pos<<endl;
        // cout<<endl;
        if(receiver[pos-1] == 1){
            receiver[pos-1] = 0;
        }else{
            receiver[pos-1] = 1;
        }
        cout<<"The corrected receiver bits are: ";
        for(int i = n-1; i>=0; i--){
            cout<<receiver[i];
        }
        cout<<endl;
    }
}
//xor for CRC
void XOR(int* total, int total_bits, int * generator, int generate_n, int
index){
    for(int i = 0; i<generate_n; i++){
        if(total[index+i] == generator[i]){
            total[index+i] = 0;
        }else{
            total[index+i] = 1;
        }
    }
}
//function for CRC
void CRC(int* total, int total_bits, int * generator, int generate_n){
    for(int i = 0; i<=total_bits - generate_n; i++){
        if(total[i] == 1){
            XOR(total, total_bits,generator,generate_n, i);
            i--;
            // cout<<"After XOR: ";
            // for(int i = 0; i<total_bits; i++){
            //     cout<<total[i];
            // }
            // cout<<endl;
        }
    }
}
void c_r_c(int sender[], int receiver[],int n)
{
     int n_generator;
    cout<<"Enter the number of bits in generator polynomial: ";
    cin>>n_generator;
    int generator[n_generator];
    cout<<"Enter the generator polynomial: ";
    for(int i = 0; i<n_generator;i++){
        cin>>generator[i];
    }
    cout<<endl;

    cout<<"The sender bits:   ";
    for(int i = 0; i<n; i++){
        cout<<sender[i];
    }
    cout<<endl;

    cout<<"The receiver bits: ";
    for(int i = 0; i<n; i++){
        cout<<receiver[i];
    }
    cout<<endl;

    int total_bits = n + n_generator -1;
    int total_sender[total_bits];
    for(int i = 0; i<n; i++){
        total_sender[i] = sender[i];
    }
    for(int i = n; i<total_bits; i++){
        total_sender[i] = 0;
    }

    // cout<<"After appending zeros to sender bits: ";
    // for(int i = 0; i<total_bits; i++){
    //     cout<<total_sender[i];
    // }
    // cout<<endl;

    CRC(total_sender, total_bits, generator, n_generator);

    // cout<<"After performing CRC on sender: ";
    // for(int i = 0; i<total_bits; i++){
    //     cout<<total_sender[i];
    // }
    // cout<<endl;
    cout<<"Check word for sender is: ";
    for(int i = 0; i<n_generator-1; i++){
        cout<<total_sender[n+i];
    }
    cout<<endl;

    cout<<"The sender bits are:   ";
    for(int i = 0; i<n; i++){
        cout<<sender[i];
    }
    for(int i = 0; i<n_generator-1; i++){
        cout<<total_sender[n+i];
    }
    cout<<endl;
```

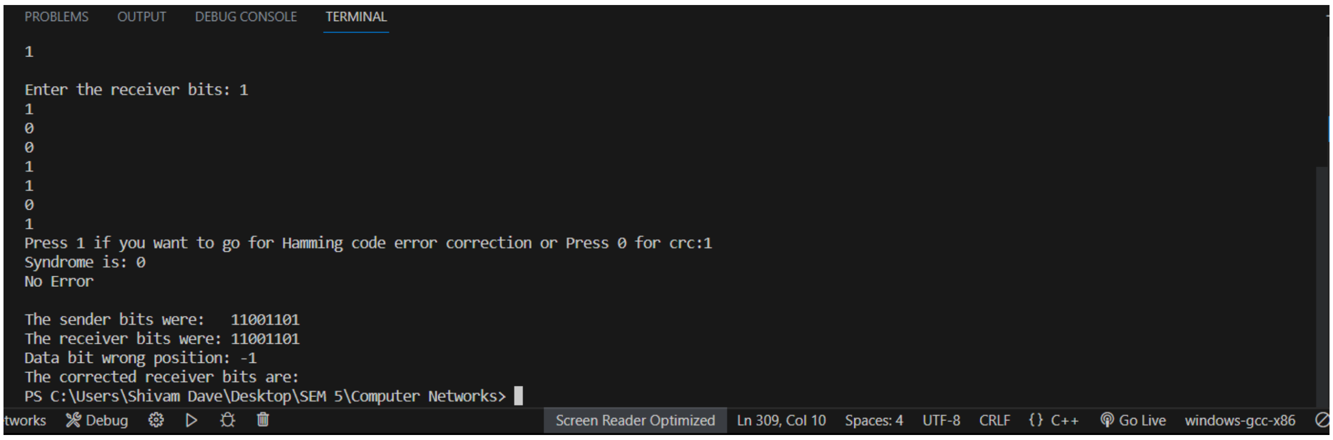
```cpp
    int total_receiver[total_bits];
    for(int i = 0; i<n; i++){
        total_receiver[i] = receiver[i];
    }
    for(int i = 0; i<n_generator; i++){
        total_receiver[n+i] = total_sender[n+i];
    }

    cout<<"The receiver bits are: ";
    for(int i = 0; i<total_bits; i++){
        cout<<total_receiver[i];
    }
    cout<<endl;

    CRC(total_receiver, total_bits, generator, n_generator);

    cout<<"After performing CRC on receiver: ";
    for(int i = 0; i<total_bits; i++){
        cout<<total_receiver[i];
    }
    cout<<endl;
    // cout<<"Check word for receiver is: ";
    bool found_error = false;
    for(int i = 0; i<n_generator; i++){
        // cout<<total_receiver[n+i-1];
        if(total_receiver[n+i-1]){
            // cout<<total_receiver[n+i-1]<<" ";
            // cout<<"Index: "<<n+i<<endl;
            found_error = true;
            break;
        }
    }

    if(found_error){
        cout<<"Error in transmission"<<endl;
    }else{
        cout<<"No error in transmission"<<endl;
    }
}

int main()
{
    int n;
    cout<<"Enter the number of bits: ";
    cin>>n;
    cout<<"Enter the sender bits: ";
    int sender[n];
    for(int i = n-1; i>=0; i--){
        cin>>sender[i];
        if(sender[i]>1 )
        {
            cout<<"You have to enter binary digits only!";
            break;
        }
    }

    cout<<"Enter the receiver bits: ";
    int receiver[n];
    for(int i = n-1; i>=0; i--){
        cin>>receiver[i];
        // if(receiver[i]!=1 || receiver[i]!=0)
        // {
        //     cout<<"You have to enter binary digits only!";
        // }
    }
    int choose;
    cout<<"Press 1 if you want to go for Hamming code error correction or Press 0 for checksum:";
    cin>>choose;
    switch(choose)
    {
        case 1:{
            hamming(sender,receiver,n);
        }
        case 0:{
            c_r_c(sender,receiver,n);
        }
        default:
        {
            cout<<"You didn't press 1 or 0";
            break;
        }
}
}
```

Output(s):

For hamming code where receiver input is same as sender input:



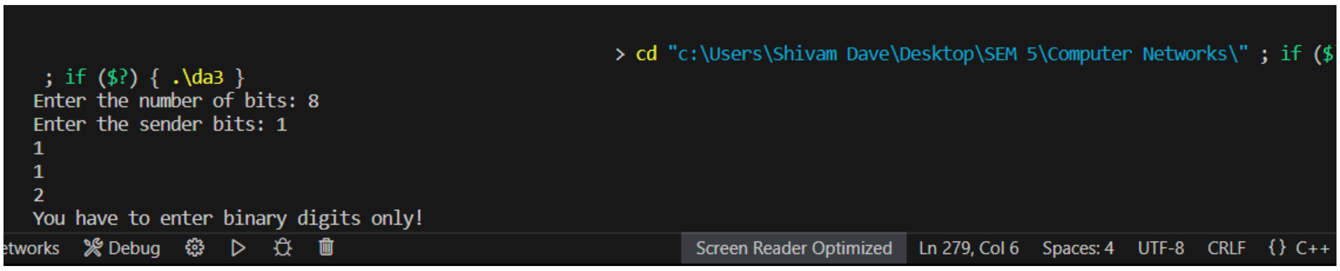
```
PS C:\Users\Shivam Dave\Desktop\SEM 5\Computer Networks> cd "c:\Users\Shivam Dave\Desktop\SEM 5\Computer Networks\" ; if ($?) { g++ da3.cpp
 -o da3 } ; if ($?) { .\da3 }
Enter the number of bits: 8
Enter the sender bits: 1
1
0
0
1
1
0
1

Enter the receiver bits: 1
1
0
0
1
1
0
```



```
1

Enter the receiver bits: 1
1
0
0
1
1
0
1
Press 1 if you want to go for Hamming code error correction or Press 0 for crc:1
Syndrome is: 0
No Error

The sender bits were:   11001101
The receiver bits were: 11001101
Data bit wrong position: -1
The corrected receiver bits are:
PS C:\Users\Shivam Dave\Desktop\SEM 5\Computer Networks> 
```

For inputting a number that is not Binary digits:

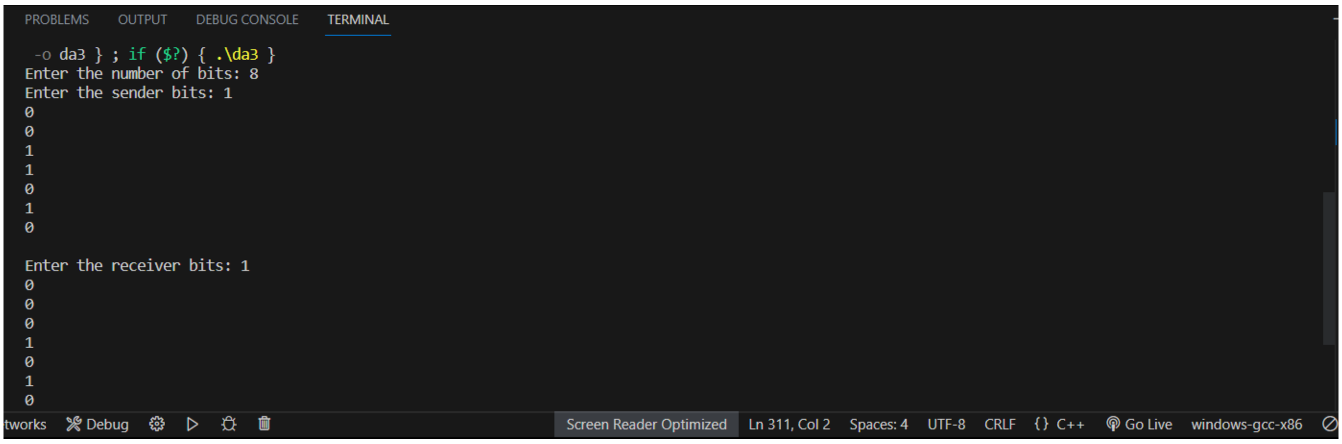
21BCB0107



```
                                                       > cd "c:\Users\Shivam Dave\Desktop\SEM 5\Computer Networks\" ; if ($
 ; if ($?) { .\da3 }
Enter the number of bits: 8
Enter the sender bits: 1
1
1
2
You have to enter binary digits only!
```

etworks Debug Screen Reader Optimized Ln 279, Col 6 Spaces: 4 UTF-8 CRLF {} C++

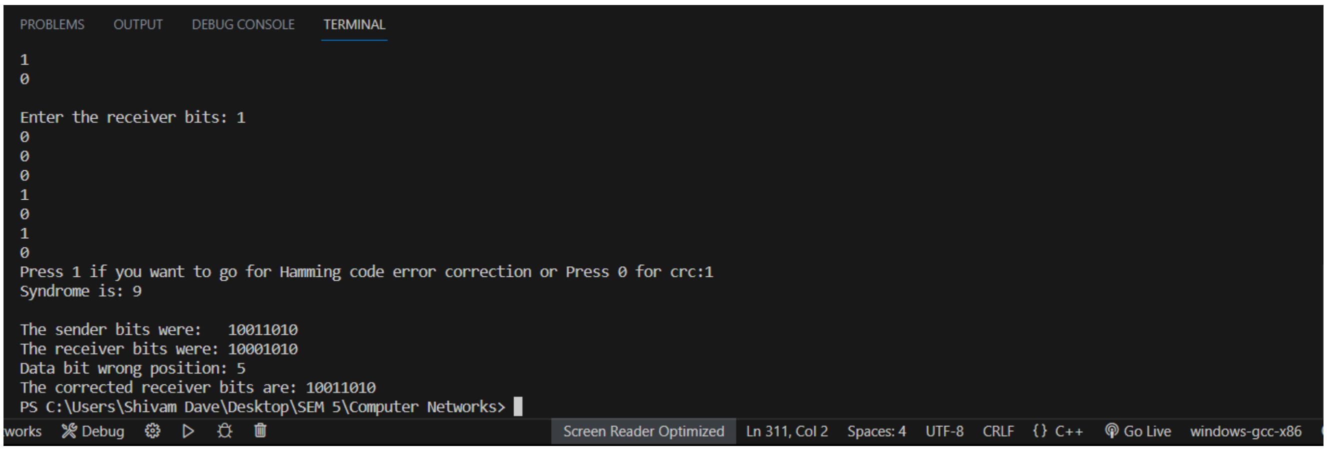
For incorrect input in receiver's side in Hamming code:



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
 -o da3 } ; if ($?) { .\da3 }
Enter the number of bits: 8
Enter the sender bits: 1
0
0
1
1
0
1
0

Enter the receiver bits: 1
0
0
0
1
0
1
0
```

tworks Debug Screen Reader Optimized Ln 311, Col 2 Spaces: 4 UTF-8 CRLF {} C++ Go Live windows-gcc-x86



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

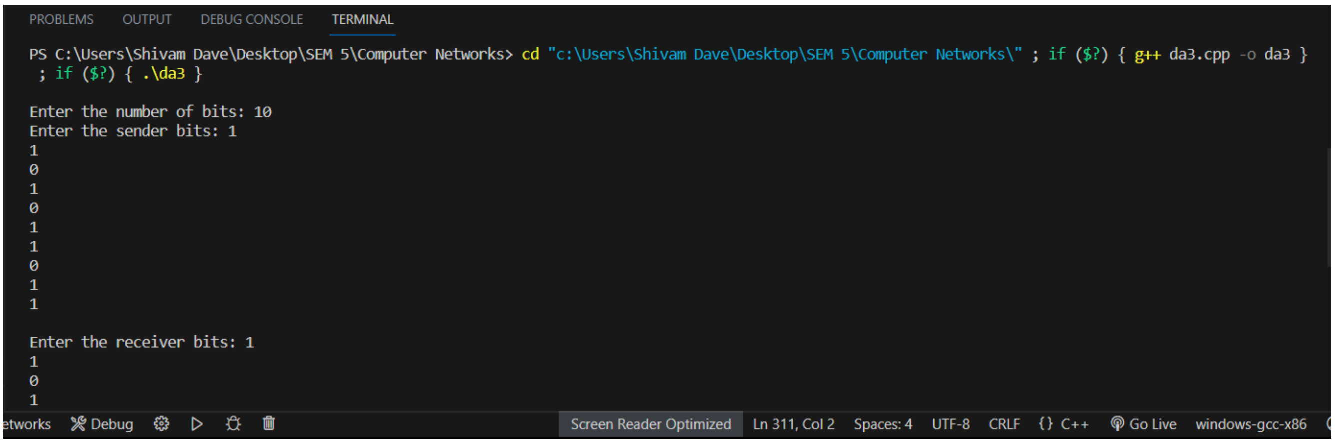
```
1
0

Enter the receiver bits: 1
0
0
0
1
0
1
0
Press 1 if you want to go for Hamming code error correction or Press 0 for crc:1
Syndrome is: 9

The sender bits were:   10011010
The receiver bits were: 10001010
Data bit wrong position: 5
The corrected receiver bits are: 10011010
PS C:\Users\Shivam Dave\Desktop\SEM 5\Computer Networks>
```

works Debug Screen Reader Optimized Ln 311, Col 2 Spaces: 4 UTF-8 CRLF {} C++ Go Live windows-gcc-x86

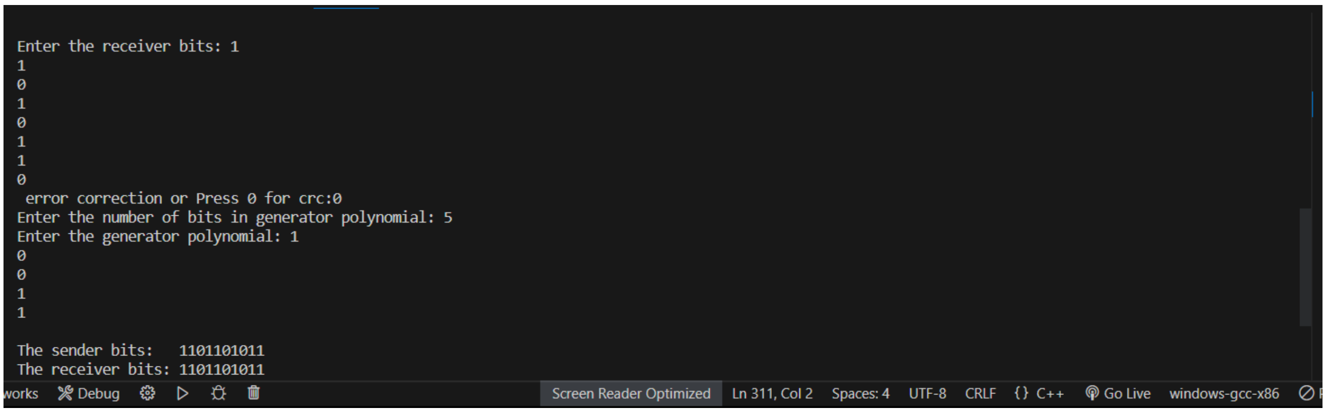
21BCB0107

For CRC:

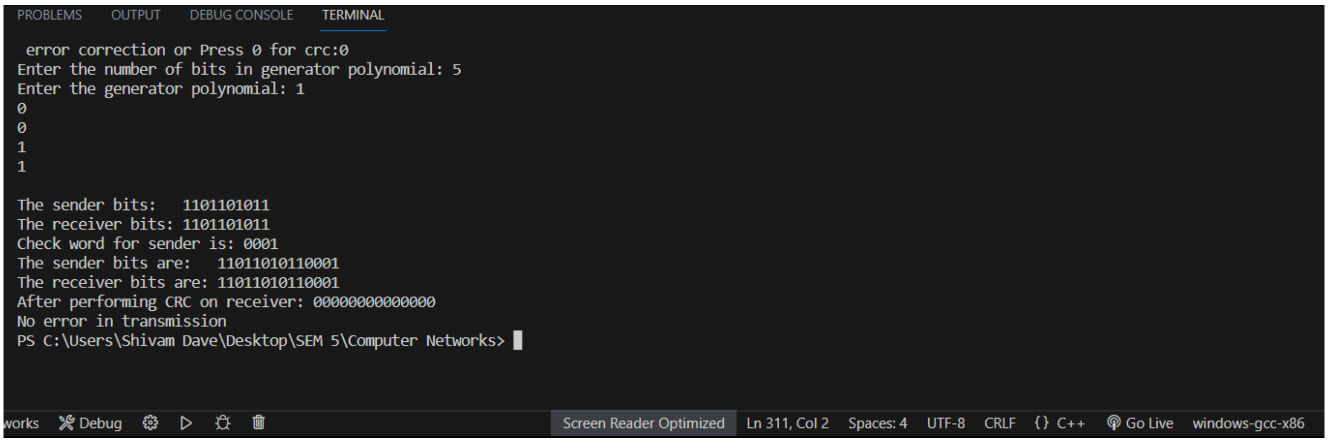
Correct receiver side’s input:



```
PS C:\Users\Shivam Dave\Desktop\SEM 5\Computer Networks> cd "c:\Users\Shivam Dave\Desktop\SEM 5\Computer Networks\" ; if ($?) { g++ da3.cpp -o da3 }
 ; if ($?) { .\da3 }

Enter the number of bits: 10
Enter the sender bits: 1
1
0
1
0
1
1
0
1
1

Enter the receiver bits: 1
1
0
1
```



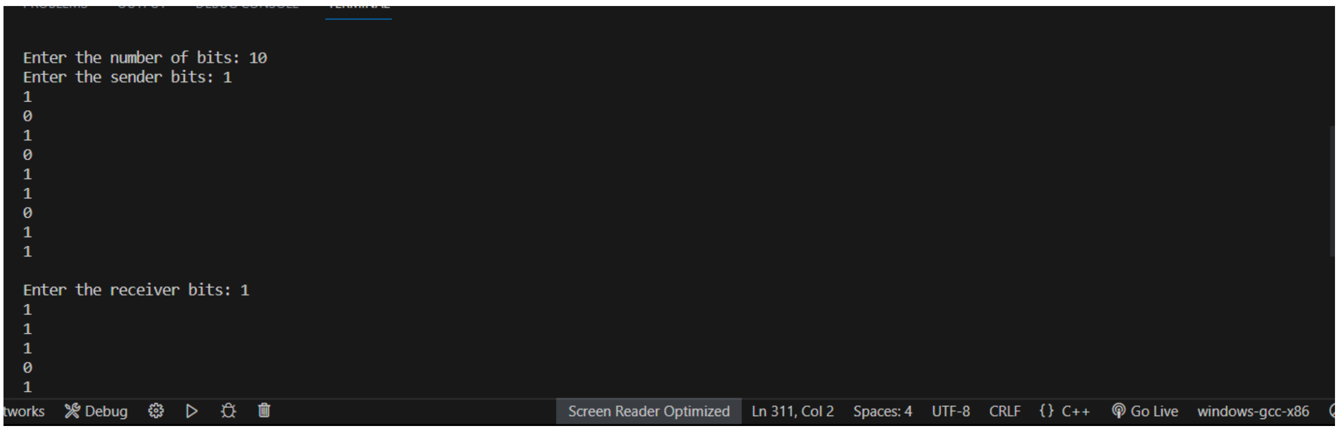
```
Enter the receiver bits: 1
1
0
1
0
1
1
0
 error correction or Press 0 for crc:0
Enter the number of bits in generator polynomial: 5
Enter the generator polynomial: 1
0
0
1
1

The sender bits:   1101101011
The receiver bits: 1101101011
```



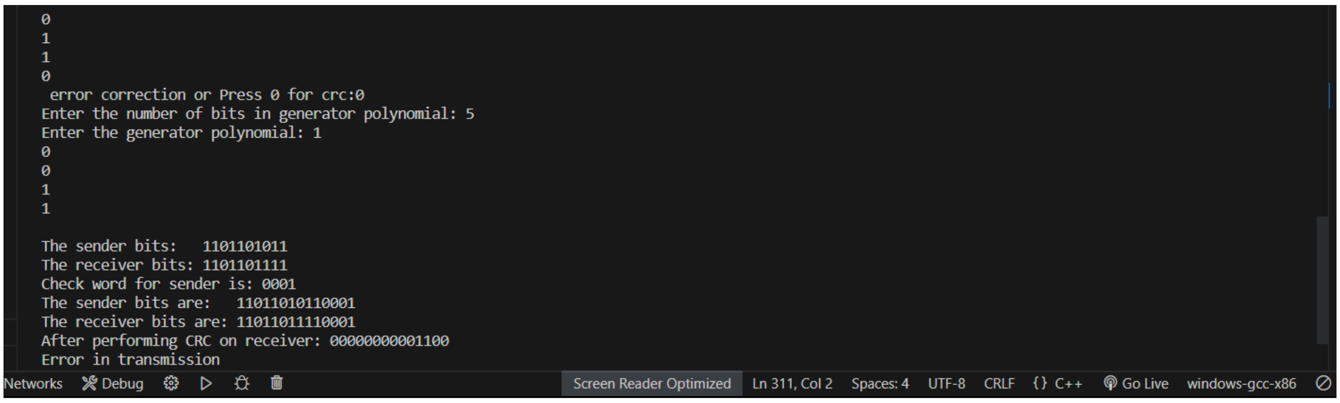
```
 error correction or Press 0 for crc:0
Enter the number of bits in generator polynomial: 5
Enter the generator polynomial: 1
0
0
1
1

The sender bits:   1101101011
The receiver bits: 1101101011
Check word for sender is: 0001
The sender bits are:   11011010110001
The receiver bits are: 11011010110001
After performing CRC on receiver: 00000000000000
No error in transmission
PS C:\Users\Shivam Dave\Desktop\SEM 5\Computer Networks> 
```

21BCB0107

For incorrect receiver side's input:



```
Enter the number of bits: 10
Enter the sender bits: 1
1
0
1
0
1
1
0
1
1

Enter the receiver bits: 1
1
1
1
0
1
```



```
0
1
1
0
 error correction or Press 0 for crc:0
Enter the number of bits in generator polynomial: 5
Enter the generator polynomial: 1
0
0
1
1

The sender bits:   1101101011
The receiver bits: 1101101111
Check word for sender is: 0001
The sender bits are:   11011010110001
The receiver bits are: 11011011110001
After performing CRC on receiver: 00000000001100
Error in transmission
```