Green University of Bangladesh
Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering

Semester: (Fall, Year:2021), B.Sc. in CSE (Day)

Assignment NO #02

Course Title: Data Communication Lab

Course Code:308 Section:193PC/DA

Lab Experiment Name: CRC and Hamming Code

Student Details

Name		ID
1.	Mohammad Nazmul Hossain	193902031
2.		
3.		

Lab Date : 14 - 12 - 21

Submission Date : 19 - 12 - 21

Course Teacher's Name : Amena Zahan

[For Teachers use only: Don't Write Anything inside this box]

Lab Report Status	
Marks: ……	**Signature:.....**
Comments:.....	**Date:.....**

1. BitStuffing and BitdeStuffing

2. OBJECTIVES/AIM [1]

Perform on In this Assignment we will solve Hamming Code and CRC Code in **c/c++.**

3. PROCEDURE / ANALYSIS / DESIGN [2]

In Hamming code:

- At first I Will take input from the user
- Then I will show the encoded code of the data
- Then I print a message for writing the encoded code
- If the Encoded code and input encoded code are matched with each other then I will show a message that "No error while transmission of data".
- Otherwise, I will print a message that the Error position, Data Sent, Data Received and the correct message.

In CRC Code:

- First I will take the input from the user for data.
- Then again I will take the input from the user for the key.
- Then I print the newly generated data and transmitted data.
- After That, I will show a message to put the transmitted data which is generated.
- If the received code contains 0 then it will print a message for "Successful".
- Otherwise, it will show a message that "Received code contains errors..."

4. IMPLEMENTATION [2]

☐ **Hamming Code**

```
#include <iostream>
#include <math.h>
#include <cstring>
using namespace std;
```

```
int check_parity(int n,int i,int code[])
{

    int p=0,k;
    for(int j=i;j<=n;j=k+i)
    {
        for(k=j;k<j+i && k<=n;k++)        //for i=1 ->bits=1,3,5,7,9,11    for i=2 ->bits 2,3,6,7,10,11
        {                    //i is parity bit position
            if(code[k]==1)
            p++;
        }
    }
    if(p%2==0)
        return 0;         //if no. of 1 is even return 0
    else
        return 1;         //if no. of 1 is odd return 1
}

void hamming_code(int data[], int num)
{
    int r=0,m=0,n,j=1,c,code[50];

    while((r+num+1)>(pow(2,r)))            //calculating no. of parity/redundant bits
        r++;

    n=num+r;                    //adding no. of redundant bits to array size
    for(int i=1;i<=n;i++)
    {
        if(i==pow(2,m) && m<=r)
        {
            code[i]=0;            //initializing all the bit position of power 2 to zero
            m++;
        }
        else
        {
            code[i]=data[j];        //assigning data to remaining positions
            j++;
        }
    }

    m=0;
```

```
    for(int i=1;i<=n;i++)
    {
        if(i==pow(2,m) && m<=r)
        {
            c=check_parity(n,i,code);        //assigning parity bit to position of power 2
            code[i]=c;
            m++;
        }
    }

    cout<<"The hamming code for given data is:";
    for(int i=n;i>0;i--)
    cout<<code[i];
    cout<<"\nEnter the received code:";
    for(int i=n;i>0;i--)
    cin>>code[i];
    m=j=c=0;
    for(int i=1;i<=n;i++)
    {
        if(i==pow(2,m) && m<=r)
        {
            c=c+(pow(2,j)*check_parity(n,i,code));    // decimal value of error code
            j++;
            m++;
        }
    }
    if(c==0)
        cout<<"\nReceived word is correct.";
    else
    {
        cout<<"\nThere is error in bit "<<(n-c)+1<<"\nThe corrected code is:";
        if(code[c]==1)
            code[c]=0;
        else
            code[c]=1;
        for(int i=n;i>0;i--)
            cout<<code[i];
    }
}

int main()
{
```

```
    int data[50], num;

    cout<<"Enter the size of data";
    cin>>num;
    cout<<"Enter the data:";
    for(int i=num;i>0;i--)
    cin>>data[i];

    hamming_code(data, num);

    return 0;
}
```

☐ CRC

```
/*
Cyclic redundancy check (CRC):
error-detecting codes which are additional data added to a given digital message to help us detect if any error has occurred during transmission of the message.
Basic approach used for error detection is the use of redundancy bits, where additional bits are added to facilitate detection of errors.
*/

#include <iostream>

using namespace std;

string xorfun(string encoded, string crc) //Bitwise XOR operation
{
    int crclen = crc.length();

    for (int i = 0; i <= (encoded.length() - crclen);) // performing bitwise xor operation
    {                                                  // " 0 xor 0 = 0"     " 1 xor 1 = 0 "
        for (int j = 0; j < crclen; j++)               // " 0 xor 1 = 1 "    " 1 xor 0 = 1"
        {
            encoded[i + j] = encoded[i + j] == crc[j] ? '0' : '1'; //if encoded bit and crc bit are same , then replace it with zero
        }
```

```
        for (; i < encoded.length() && encoded[i] != '1'; i++)
            ;
    }

    return encoded;
}

int main()
{
    string data, crc, encoded = "";
    cout << endl
         << "-----------Sender Side --------------" << endl;
    cout << "Enter Data bits: " << endl;
    cin >> data; //data bits need to be transmitted

    cout << "Enter Generator: " << endl;
    cin >> crc; //crc - genearator polynomial ( agreed by sender & reciever)

    encoded += data; //encoded bits are initialized to data bits

    int datalen = data.length();
    int crclen = crc.length();

    for (int i = 1; i <= (crclen - 1); i++)
        encoded += '0'; //appending length of (generator polinomial -1) number of zeros to encoded bits

    encoded = xorfun(encoded, crc); //performing bitwise xor to obtain

    cout << "Checksum generated is: ";
    cout << encoded.substr(encoded.length() - crclen + 1) << endl
         << endl; //data bits + checksum bit is what going to be sent to reciever
    cout << "Message to be Transmitted over network: ";
    cout << data + encoded.substr(encoded.length() - crclen + 1); //this is the message going to be sent to
the Reciever

    cout << endl
         << "---------------Reciever Side-----------------" << endl;

    cout << "Enter the message recieved: " << endl;
    string msg; //Reciever enters the recieved message
    cin >> msg;
```

```
    msg = xorfun(msg, crc); //bitwise xor is performed between recieved bits and the generator crc bits

    for (char i : msg.substr(msg.length() - crclen + 1)) //after performing xor , if the last few bits are zero
then there's no error in transmission
        if (i != '0')
        {
            cout << "Error in communication " << endl; //if bits not zero ; ERROR IN TRANSMISSION
            return 0;
        }

    cout << "No Error !" << endl; //else NO ERROR
    return 0;
}
```

5. TEST RESULT / OUTPUT [2]

Output - Hamming Code in C++

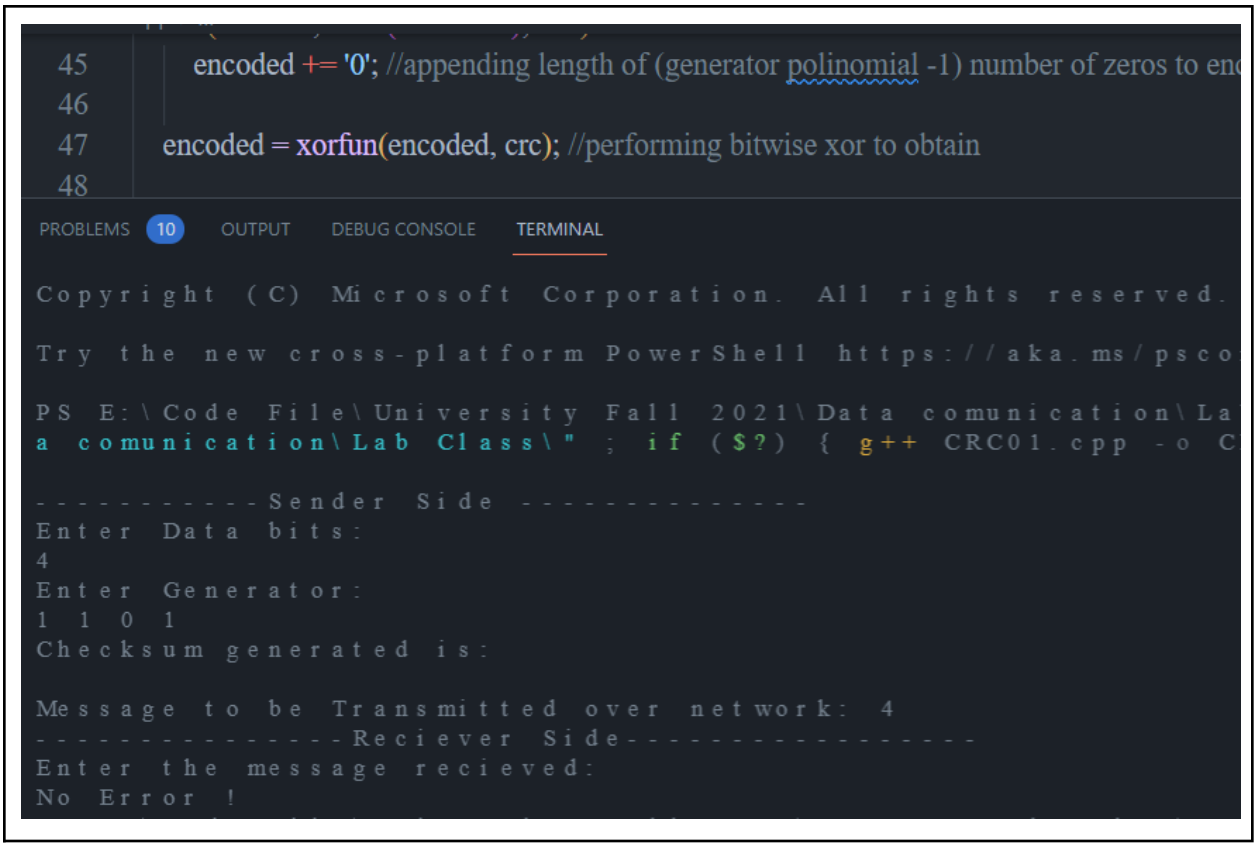
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights rese

Try the new cross-platform PowerShell https://aka.ms

PS E:\Code File\University Fall 2021\Data comunicati
lass\" ; if ($?) { g++ hammingCode01.cpp -o hammingC
Enter the size of data4
Enter the data:0 1 1 1
The hamming code for given data is:0110100
Enter the received code:0 1 1 0 1 0 0

Received word is correct.
```

Output - CRC in C++


45 encoded += '0'; //appending length of (generator polinomial -1) number of zeros to enc
46
47 encoded = xorfun(encoded, crc); //performing bitwise xor to obtain
48
PROBLEMS 10 OUTPUT DEBUG CONSOLE TERMINAL
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell https://aka.ms/psco
PS E:\Code File\University Fall 2021\Data comunication\La
a comunication\Lab Class\" ; if ($?) { g++ CRC01.cpp -o C
----------Sender Side ----------
Enter Data bits:
4
Enter Generator:
1 1 0 1
Checksum generated is:
Message to be Transmitted over network: 4
----------Reciever Side----------
Enter the message recieved:
No Error !