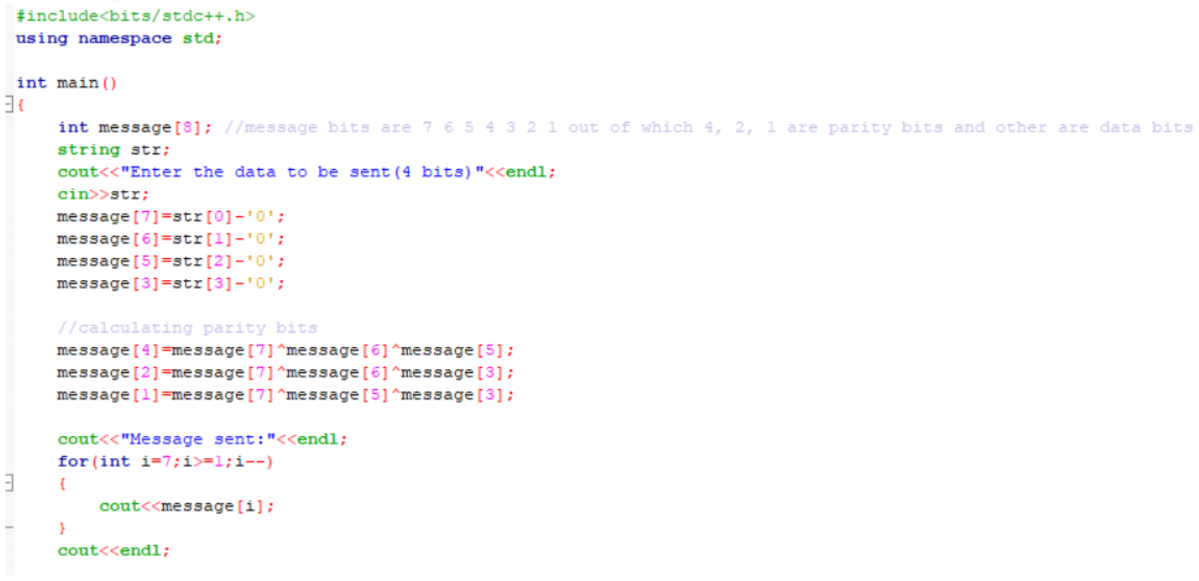
CS359: ASSIGNMENT-7

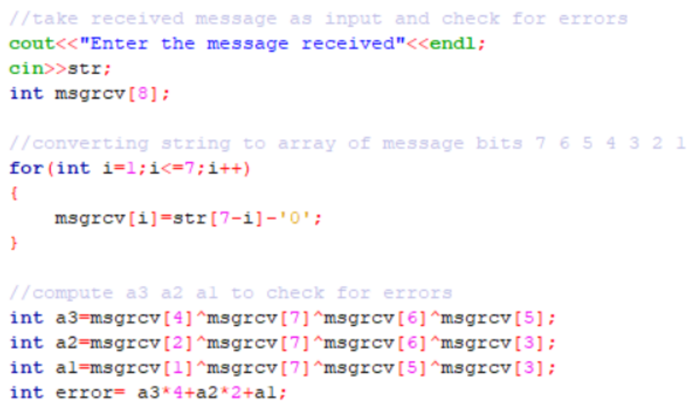
1901CS75

SIDDHARTH SANSKRITAYAN

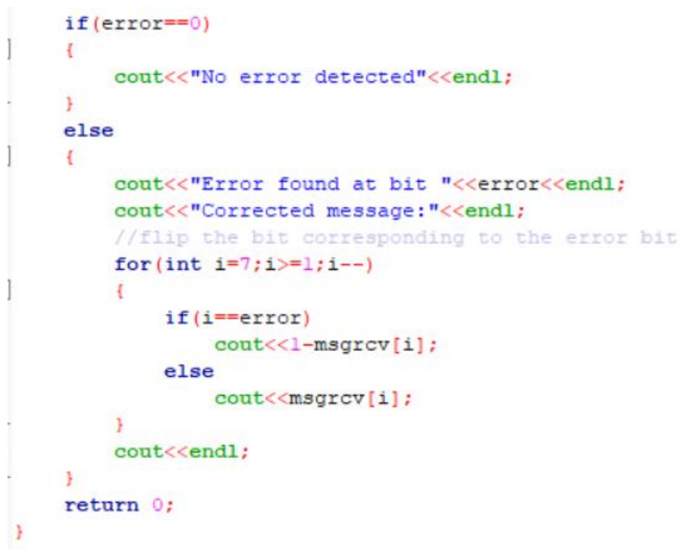
- Hamming code:



```
#include<bits/stdc++.h>
using namespace std;

int main()
{
    int message[8]; //message bits are 7 6 5 4 3 2 1 out of which 4, 2, 1 are parity bits and other are data bits
    string str;
    cout<<"Enter the data to be sent(4 bits)"<<endl;
    cin>>str;
    message[7]=str[0]-'0';
    message[6]=str[1]-'0';
    message[5]=str[2]-'0';
    message[3]=str[3]-'0';

    //calculating parity bits
    message[4]=message[7]^message[6]^message[5];
    message[2]=message[7]^message[6]^message[3];
    message[1]=message[7]^message[5]^message[3];

    cout<<"Message sent:"<<endl;
    for(int i=7;i>=1;i--)
    {
        cout<<message[i];
    }
    cout<<endl;
```

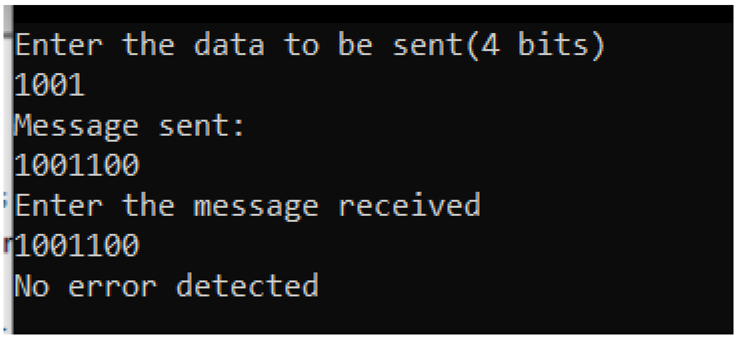


```
    //take received message as input and check for errors
    cout<<"Enter the message received"<<endl;
    cin>>str;
    int msgrcv[8];

    //converting string to array of message bits 7 6 5 4 3 2 1
    for(int i=1;i<=7;i++)
    {
        msgrcv[i]=str[7-i]-'0';
    }

    //compute a3 a2 a1 to check for errors
    int a3=msgrcv[4]^msgrcv[7]^msgrcv[6]^msgrcv[5];
    int a2=msgrcv[2]^msgrcv[7]^msgrcv[6]^msgrcv[3];
    int a1=msgrcv[1]^msgrcv[7]^msgrcv[5]^msgrcv[3];
    int error= a3*4+a2*2+a1;
```

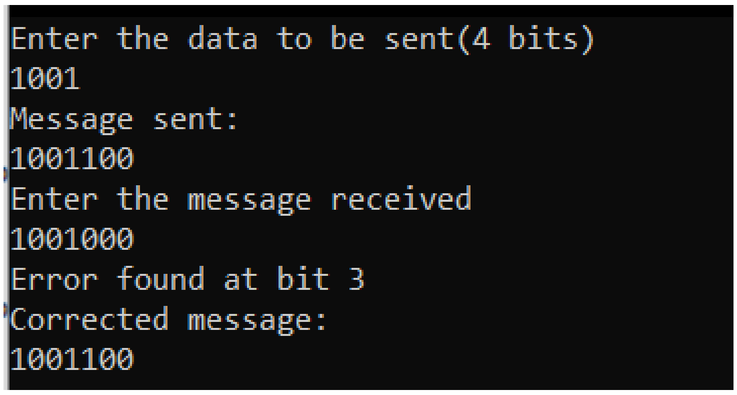


```
    if(error==0)
    {
        cout<<"No error detected"<<endl;
    }
    else
    {
        cout<<"Error found at bit "<<error<<endl;
        cout<<"Corrected message:"<<endl;
        //flip the bit corresponding to the error bit
        for(int i=7;i>=1;i--)
        {
            if(i==error)
                cout<<1-msgrcv[i];
            else
                cout<<msgrcv[i];
        }
        cout<<endl;
    }
    return 0;
}
```

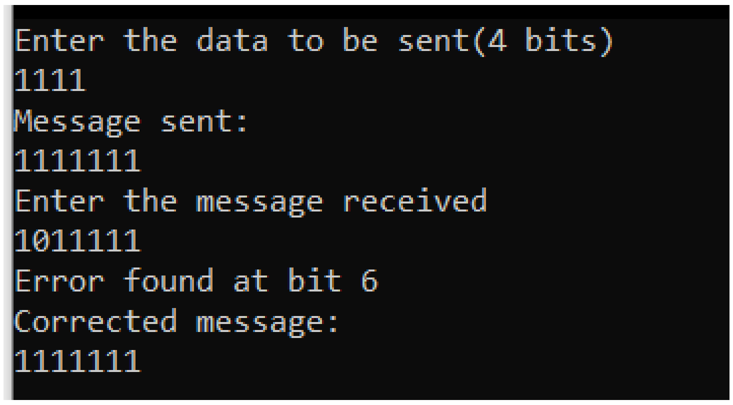
Sample Test Cases:



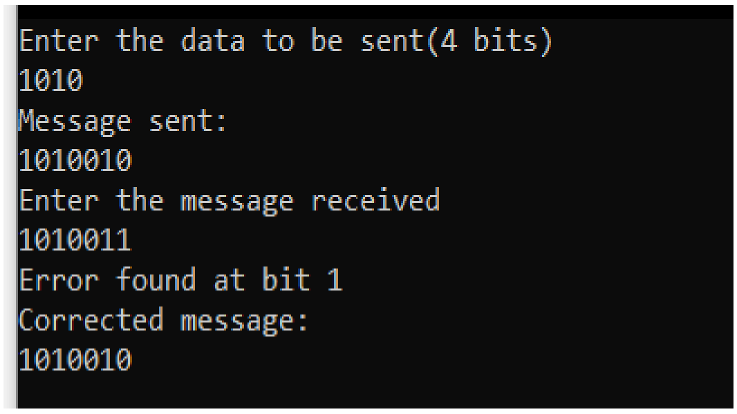
```
Enter the data to be sent(4 bits)
1001
Message sent:
1001100
Enter the message received
1001100
No error detected
```



```
Enter the data to be sent(4 bits)
1001
Message sent:
1001100
Enter the message received
1001000
Error found at bit 3
Corrected message:
1001100
```

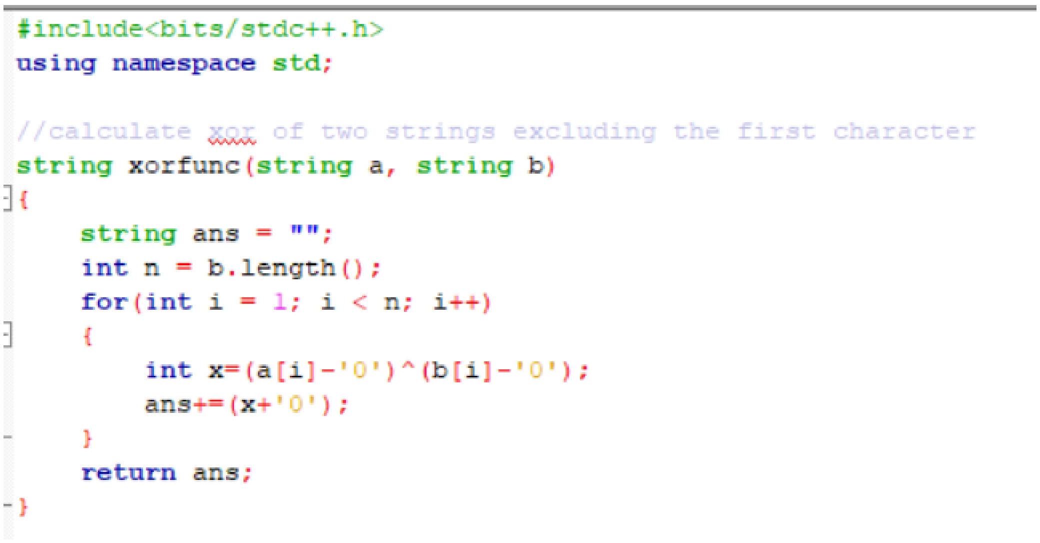


```
Enter the data to be sent(4 bits)
1111
Message sent:
1111111
Enter the message received
1011111
Error found at bit 6
Corrected message:
1111111
```

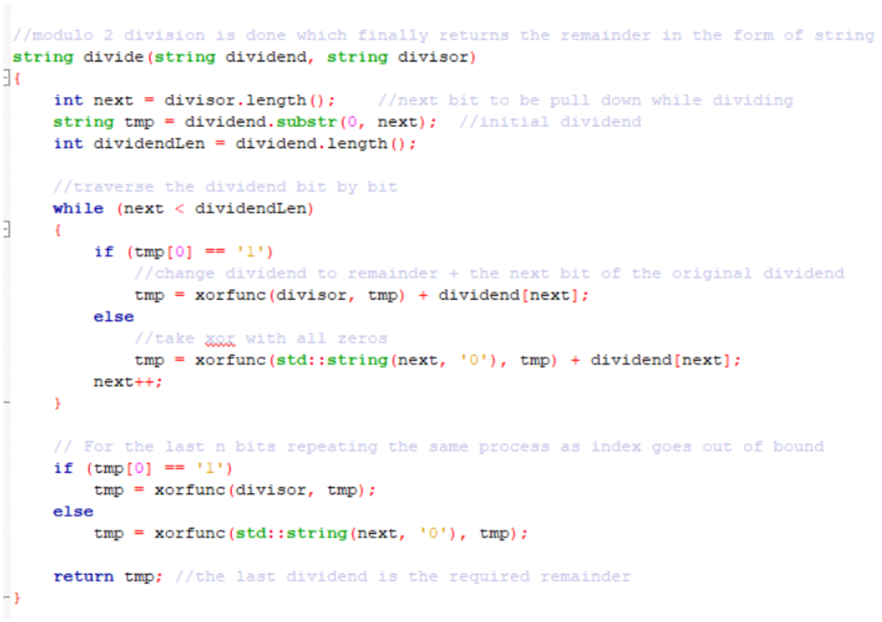


```
Enter the data to be sent(4 bits)
1010
Message sent:
1010010
Enter the message received
1010011
Error found at bit 1
Corrected message:
1010010
```

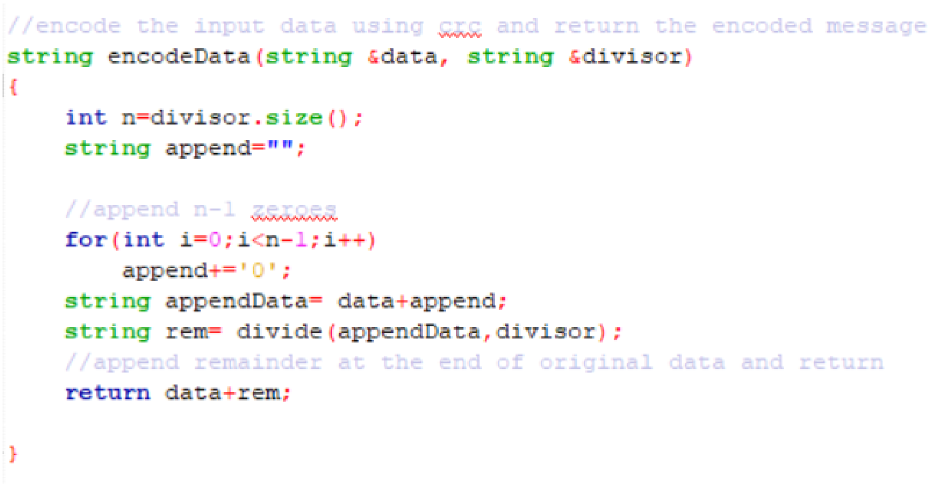
- CRC Code:



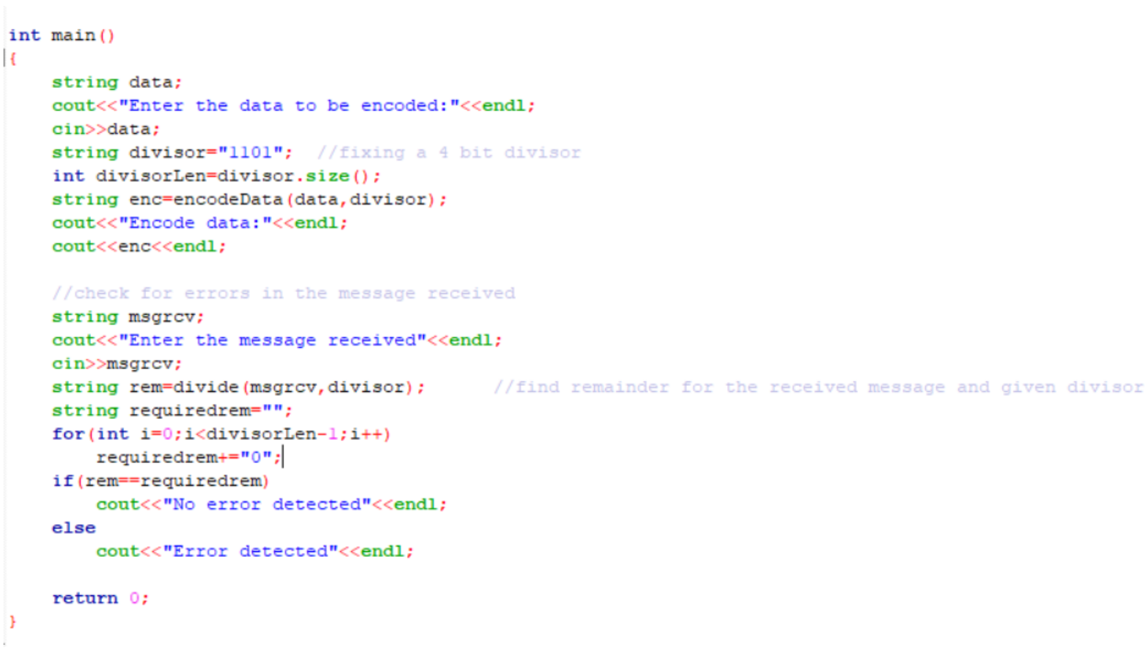
```cpp
#include<bits/stdc++.h>
using namespace std;

//calculate xor of two strings excluding the first character
string xorfunc(string a, string b)
{
    string ans = "";
    int n = b.length();
    for(int i = 1; i < n; i++)
    {
        int x=(a[i]-'0')^(b[i]-'0');
        ans+=(x+'0');
    }
    return ans;
}
```



```cpp
//modulo 2 division is done which finally returns the remainder in the form of string
string divide(string dividend, string divisor)
{
    int next = divisor.length();     //next bit to be pull down while dividing
    string tmp = dividend.substr(0, next);  //initial dividend
    int dividendLen = dividend.length();

    //traverse the dividend bit by bit
    while (next < dividendLen)
    {
        if (tmp[0] == '1')
            //change dividend to remainder + the next bit of the original dividend
            tmp = xorfunc(divisor, tmp) + dividend[next];
        else
            //take xor with all zeros
            tmp = xorfunc(std::string(next, '0'), tmp) + dividend[next];
        next++;
    }

    // For the last n bits repeating the same process as index goes out of bound
    if (tmp[0] == '1')
        tmp = xorfunc(divisor, tmp);
    else
        tmp = xorfunc(std::string(next, '0'), tmp);

    return tmp; //the last dividend is the required remainder
}
```

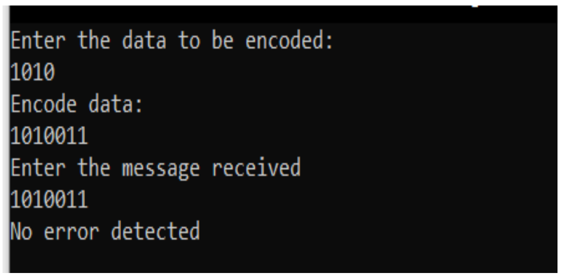


```cpp
//encode the input data using crc and return the encoded message
string encodeData(string &data, string &divisor)
{
    int n=divisor.size();
    string append="";

    //append n-1 zeroes
    for(int i=0;i<n-1;i++)
        append+='0';
    string appendData= data+append;
    string rem= divide(appendData,divisor);
    //append remainder at the end of original data and return
    return data+rem;

}
```

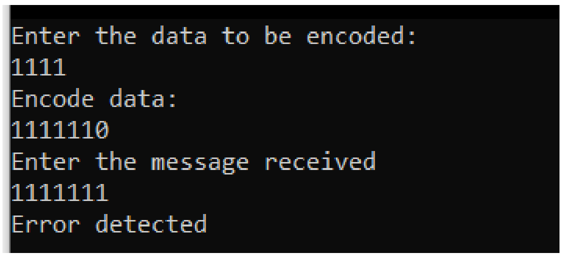


```
int main()
{
    string data;
    cout<<"Enter the data to be encoded:"<<endl;
    cin>>data;
    string divisor="1101";  //fixing a 4 bit divisor
    int divisorLen=divisor.size();
    string enc=encodeData(data,divisor);
    cout<<"Encode data:"<<endl;
    cout<<enc<<endl;

    //check for errors in the message received
    string msgrcv;
    cout<<"Enter the message received"<<endl;
    cin>>msgrcv;
    string rem=divide(msgrcv,divisor);      //find remainder for the received message and given divisor
    string requiredrem="";
    for(int i=0;i<divisorLen-1;i++)
        requiredrem+="0";
    if(rem==requiredrem)
        cout<<"No error detected"<<endl;
    else
        cout<<"Error detected"<<endl;

    return 0;
}
```

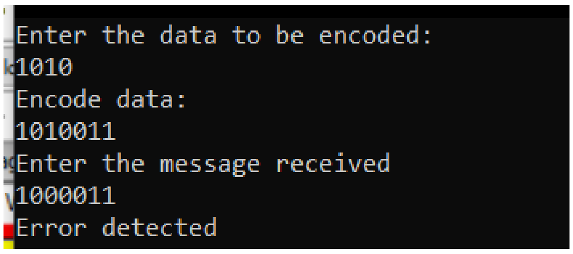
Sample Test Cases:



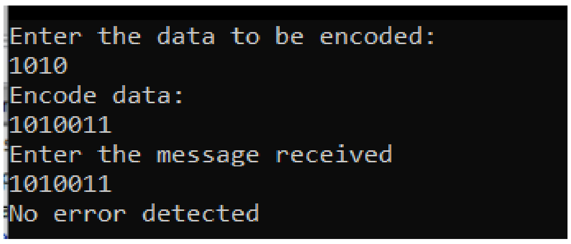
```
Enter the data to be encoded:
1010
Encode data:
1010011
Enter the message received
1010011
No error detected
```



```
Enter the data to be encoded:
1111
Encode data:
1111110
Enter the message received
1111111
Error detected
```



```
Enter the data to be encoded:
1010
Encode data:
1010011
Enter the message received
1000011
Error detected
```



```
Enter the data to be encoded:
1010
Encode data:
1010011
Enter the message received
1010011
No error detected
```