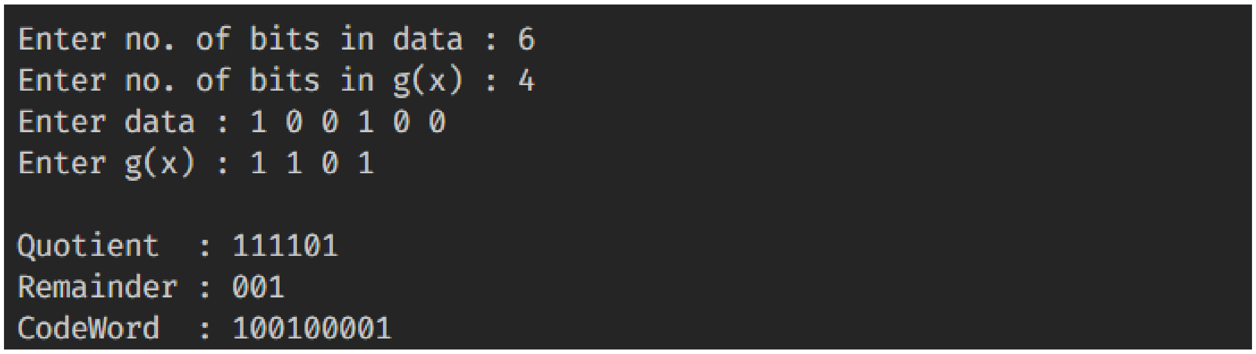
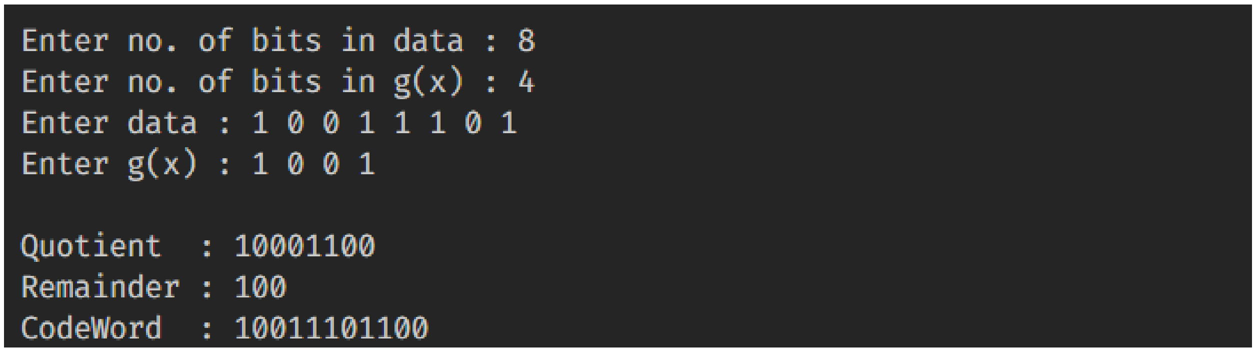
MODERN COMPUTER NETWORK LAB EXPERIMENTs

```
// SENDER SIDE Algorithm :

#include <iostream>
#define MAX 20
using namespace std;

void XOR(int num1[MAX], int num2[MAX], int result[MAX], int n) {
    for (int i=0; i<n; i++) {
        result[i] = (num1[i] == num2[i]) ? 0 : 1;
    }
}
void binaryDivision(int dividend[MAX], int divisor[MAX], int m, int n) {
    string quotient = "";
    int start = 0;
    int end = n;

    int zero[MAX] = {0};
    int result[MAX] = {0};

    for(int i=0; i<n; i++) {
        result[i] = dividend[i];
    }
    for(int i=0; i<m-n+1; i++)
    {
        int firstDigit = result[0];
        if(firstDigit == 1) {
            quotient.push_back('1');
            XOR(result, divisor, result, n);
            int nextFromDividend = dividend[end++];

            for(int i=0; i<n-1; ++i) {
                result[i] = result[i + 1];
            }
            result[n-1] = nextFromDividend;
        }
        else {
            quotient.push_back('0');
            XOR(result, zero, result, n);
            int nextFromDividend = dividend[end++];

            for(int i=0; i<n-1; ++i) {
                result[i] = result[i + 1];
            }
            result[n-1] = nextFromDividend;
        }
    }
    cout << "\nQuotient  : " << quotient << endl;
    cout << "Remainder : ";
    for(int i=0; i<n-1; i++) {
        cout << result[i];
    }
    for(int i=m-n+1 /* since we added n-1 bits */, j=0; j<m+n-1 &&
i<m+n-1; i++, j++) {
        dividend[i] = result[j]; // replacing the 'n-1' 0's appended by
remainder obtain...
    }
    cout << "\nCodeWord  : "; // displaying the resultant CodeWord....
    for(int i=0; i<m; i++) { cout << dividend[i]; }
}

int main() {
    int m, n;
    int goX[MAX];
    int data[MAX] {0};
    cout << "\nEnter no. of bits in data : "; cin >> m;
    cout << "Enter no. of bits in g(x) : "; cin >> n;
    cout << "Enter data : ";
    for(int i=0; i<m; i++) {
        cin >> data[i];
    }
    cout << "Enter g(x) : ";
    for(int i=0; i<n; i++) {
        cin >> goX[i];
    }
    for(int i=m; i<m+n-1; i++) { // appending 'n-1' 0's to data...
        data[i] = 0;
    }
    binaryDivision(data, goX, m+n-1, n);
    return 0;
}
```

OUTPUT :



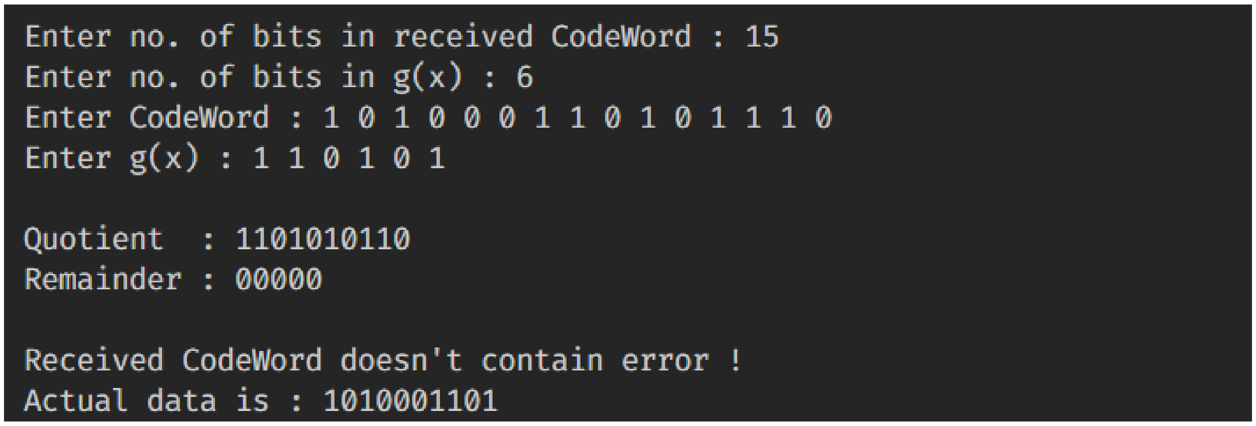
```
Enter no. of bits in data : 6
Enter no. of bits in g(x) : 4
Enter data : 1 0 0 1 0 0
Enter g(x) : 1 1 0 1

Quotient  : 111101
Remainder : 001
CodeWord  : 100100001
```



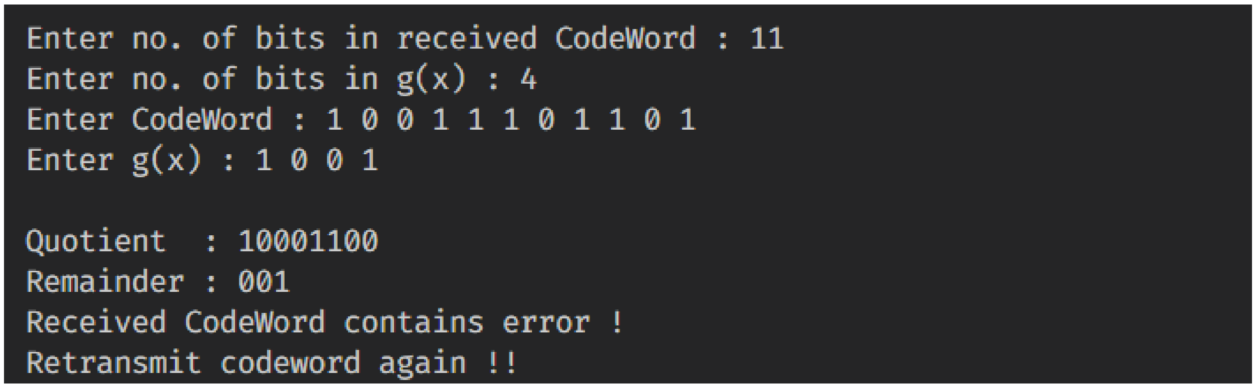
```
Enter no. of bits in data : 8
Enter no. of bits in g(x) : 4
Enter data : 1 0 0 1 1 1 0 1
Enter g(x) : 1 0 0 1

Quotient  : 10001100
Remainder : 100
CodeWord  : 10011101100
```

```
#include <iostream>
#define MAX 20
using namespace std;

void XOR(int num1[MAX], int num2[MAX], int result[MAX], int n) {
    for (int i=0; i<n; i++) {
        result[i] = (num1[i] == num2[i]) ? 0 : 1;
    }
}

void CRC_Receiver(int dividend[MAX], int divisor[MAX], int m, int n) {
    string quotient = "";
    int start = 0;
    int end = n;

    int zero[MAX] = {0};
    int result[MAX] = {0};

    for(int i=0; i<n; i++) {
        result[i] = dividend[i];
    }
    for(int i=0; i<m-n+1; i++) {
        int firstDigit = result[0];
        if(firstDigit == 1) {
            quotient.push_back('1');
            XOR(result, divisor, result, n);
            int nextFromDividend = dividend[end++];

            for(int i=0; i<n-1; ++i) {
                result[i] = result[i + 1];
            }
            result[n-1] = nextFromDividend;
        }
        else {
            quotient.push_back('0');
            XOR(result, zero, result, n);
            int nextFromDividend = dividend[end++];

            for(int i=0; i<n-1; ++i) {
                result[i] = result[i + 1];
            }
            result[n-1] = nextFromDividend;
        }
    }
    cout << "\nQuotient  : " << quotient << endl;
    cout << "Remainder : ";
    for(int i=0; i<n-1; i++) {
        cout << result[i];
    }
    bool flag = true;
    for(int i=0; i<n-1; i++) {
        if(result[i] == 1) {
            flag = false;
        }
    }
    if(flag) {
        cout << "\n\nReceived CodeWord doesn't contain error !";
        cout << "\nActual data is : ";
        for(int i=0; i<m-n+1; i++) {
            cout << dividend[i];
        }
    } else { cout << "\nReceived CodeWord contains error !\nRetransmit
codeword again !!"; }
}
int main() {
    int m, n;
    int goX[MAX];
    int code_word[MAX] {0};
    cout << "\nEnter no. of bits in received CodeWord : "; cin >> m;
    cout << "Enter no. of bits in g(x) : "; cin >> n;
    cout << "Enter CodeWord : ";
    for(int i=0; i<m; i++) {
        cin >> code_word[i];
    }
    cout << "Enter g(x) : ";
    for(int i=0; i<n; i++) {
        cin >> goX[i];
    }
    CRC_Receiver(code_word, goX, m, n);
    return 0;
}
```

OUTPUT :



```
Enter no. of bits in received CodeWord : 15
Enter no. of bits in g(x) : 6
Enter CodeWord : 1 0 1 0 0 0 1 1 0 1 0 1 1 1 0
Enter g(x) : 1 1 0 1 0 1

Quotient  : 1101010110
Remainder : 00000

Received CodeWord doesn't contain error !
Actual data is : 1010001101
```



```
Enter no. of bits in received CodeWord : 11
Enter no. of bits in g(x) : 4
Enter CodeWord : 1 0 0 1 1 1 0 1 1 0 1
Enter g(x) : 1 0 0 1

Quotient  : 10001100
Remainder : 001
Received CodeWord contains error !
Retransmit codeword again !!
```