Computer Networks (CS303)

Assignment - 4

U19CS012

1. Implement **ERROR DETECTION** technique CRC in **C++** PROGRAMMING.

File	Purpose
sender.cpp	Accept **Data** and **Key** both as Input in Binary and **Encoded Data** (Data + Checksum) as Output.
receiver.cpp	Accept **Encoded Data** (Data + Checksum) and **Key** as Input & "Error Detected" OR "Error not Detected" Output message.

What is Cyclic Redundancy Check?

Cyclic Redundancy Check CRC is an **Error Detection Algorithm** used in Communication Networks to **Check** if the Transmitted data contains any Error.

How CRC works?

Sender's Side:

1. CRC used N bit generator polynomial which works as **Divisor**.

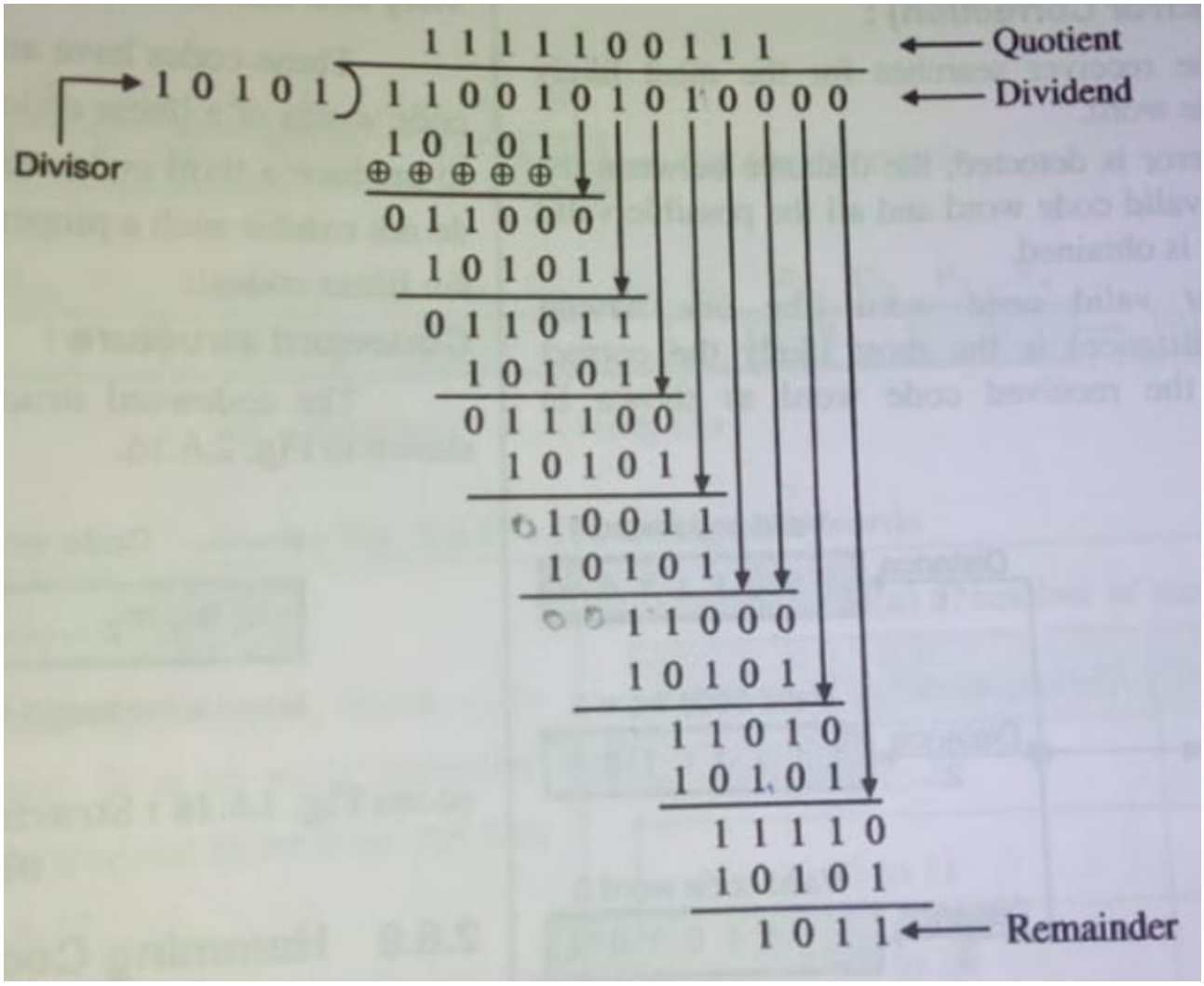
   Generator = 10101, then N = 5

2. Append **N-1** Number of zeros to the Data word.

   Data word = 110010101

   Appended Data word = 110010101 + 0000 = 1100101010000

3. Divide the appended data word by the generator by using Binary Division.



4. The remainder obtained after division is N-1 bit CRC code.

Remainder = N-1 bit CRC code = 1011

5. Replace the N-1 zeros in Data word with the N-1 bit CRC code.

Final data word = 110010101 + 1011 = 1100101011011

6. Transmit the CRC appended data word.

Receiver's end:

1. Divide the received data word by the same generator.

2. If the remainder is **Zero** than data is not Erroneous else it contains Error.

Code Implementation

```cpp
// U19CS012 Bhagya Vinod Rana

#include <iostream>
using namespace std;

// This Function return the Remainder after Binary Division
string binary_division(string encoded, string crc);

// This Function Check if the Message/CRC is Valid
bool is_valid(string s);

int main()
{
    string data, key, encoded;

    cout << "\n~~~~~~~~~~~~~~~~~~~~~ Sender Side ~~~~~~~~~~~~~~~~~~~~\n\n";

    // Data Bits that needs to be transmitted
    cout << "Enter Data Bits [0/1]: \n";

    do
    {
        cin >> data;
    } while (!is_valid(data));

    // 1.) Key used N bit generator polynomial which works as Divisor ( Agreed by Both Sender
 & Reciever)
    cout << "Enter Key [Generator] [In Binary Form, Eg. x^2 + 1 -> 101 ]: \n";

    do
    {
        cin >> key;
    } while (!is_valid(key));

    int key_len = key.length();

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

    // 2.) Appending "(length of Generator Polynomial) - 1" number of zeros to encoded bits t
o the Data word.
    for (int i = 1; i <= (key_len - 1); i++)
        encoded += '0';

    // 3. Divide the appended data word by the generator by using Binary Division.
    string CRC = binary_division(encoded, key);

    // 4. The remainder obtained after division is N-1 bit CRC code.
```

```
    cout << "\nCRC Bits Generated is: " << CRC << endl;

    // Data bits + CRC bit is What going to be sent to reciever
    cout << "Message to be Transmitted over Network: " << data + CRC << endl;

    cout << "\n~~~~~~~~~~~~~~~~~~~~~ Receiver Side ~~~~~~~~~~~~~~~~~~~~~\n\n";

    cout << "Enter the Message Recieved: " << endl;

    string msg;
    cin >> msg;

    // Remainder After Binary Division
    string rem = binary_division(msg, key);

    for (char i : rem)
    {
        if (i != '0')
        {
            cout << "\nError Detected!\n";
            return 0;
        }
    }

    cout << "\nError Not Detected!\n";
    return 0;
}

// This Function Check if the Message/CRC is Valid
bool is_valid(string s)
{
    for (auto ch : s)
    {
        if (ch != '0' && ch != '1')
        {
            cout << "Enter a Binary String!\n";
            return false;
        }
    }
    return true;
}

// This Function return the Remainder after Binary Division
string binary_division(string encoded, string key)
{
    int key_len = key.length();

    for (int i = 0; i <= (encoded.length() - key_len);)
    {
        for (int j = 0; j < key_len; j++)
```

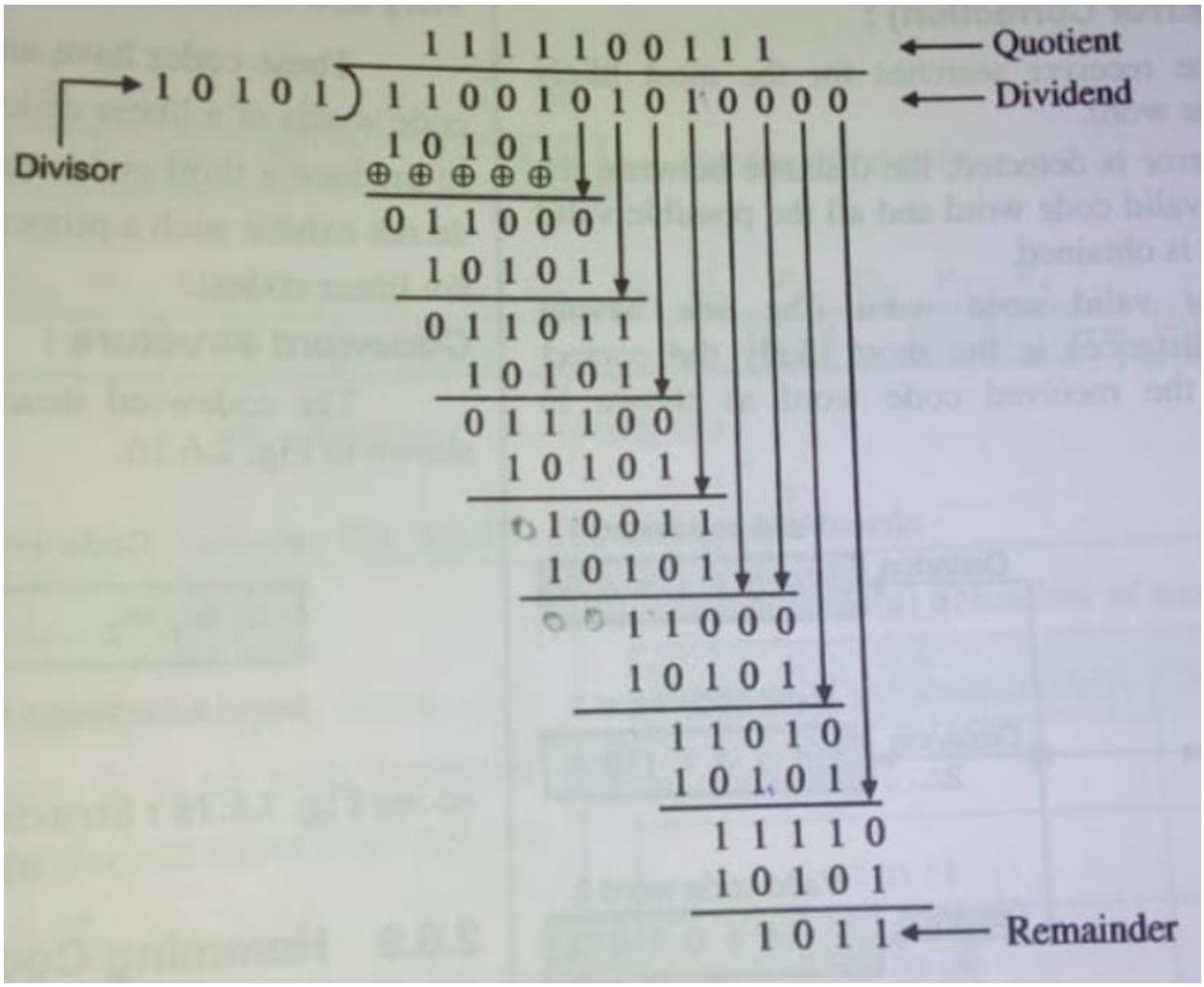
```
        {
            // if Encoded bit and CRC bit are same, then replace it with zero
            // "0 xor 0 = 0"      "1 xor 1 = 0"
            // "0 xor 1 = 1 "     "1 xor 0 = 1"
            encoded[i + j] = encoded[i + j] == key[j] ? '0' : '1';
        }

        // If Prefix are '0' we increment the Window
        for (; i < encoded.length() && encoded[i] != '1'; i++)
            ;
    }

    // Remainder Last Bits
    string rem = encoded.substr(encoded.length() - key_len + 1);
    return rem;
}
```

Test Case

Data Bits: 110010101 & Key: 10101



```
~~~~~~~~~~~~~~~~~~~~~~~~ Sender Side ~~~~~~~~~~~~~~~~~~~~~~~~

Enter Data Bits [0/1]:
110010101
Enter Key [Generator] [In Binary Form, Eg. x^2 + 1 -> 101 ]:
10101

CRC Bits Generated is: 1011
Message to be Transmitted over Network: 1100101011011

~~~~~~~~~~~~~~~~~~~~~~~~ Receiver Side ~~~~~~~~~~~~~~~~~~~~~~~~

Enter the Message Recieved:
1100101011011

Error Not Detected!
PS C:\Users\Admin\Desktop\CN_L4\CRC> cd "c:\Users\Admin\Deskto

~~~~~~~~~~~~~~~~~~~~~~~~ Sender Side ~~~~~~~~~~~~~~~~~~~~~~~~

Enter Data Bits [0/1]:
110010101
Enter Key [Generator] [In Binary Form, Eg. x^2 + 1 -> 101 ]:
10101

CRC Bits Generated is: 1011
Message to be Transmitted over Network: 1100101011011

~~~~~~~~~~~~~~~~~~~~~~~~ Receiver Side ~~~~~~~~~~~~~~~~~~~~~~~~

Enter the Message Recieved:
1111111100100

Error Detected!
PS C:\Users\Admin\Desktop\CN_L4\CRC>
```

2. Implement ERROR DETECTION technique 16-bit Checksum in **C++** PROGRAMMING.

File	Purpose
sender.cpp	Accept **Input String** (eg. Forouzan) and **Encoded String** (Input data + checksum) as Output.
receiver.cpp	Accept **Encoded Data** (Data + Checksum) and **Key** as Input & "Error Detected" OR "Error not Detected" Output message.

Code Implementation [Sender Side]

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

// Function to Convert the Number to Binary Form
vi get_binary(int n);

// 1's Complement Addition [Given there Binary Forms]
vi add_binary(vi num1, vi num2);

int main()
{
    cout << "\n~~~~~~~~~~~~~~~~~~~~~~ Sender Side ~~~~~~~~~~~~~~~~~~~~~~\n\n";
    cout << "Enter Message to be Encoded (string) : ";

    string s;
    cin >> s;

    int n = s.length();
    vi v[n];

    // Step 1 : Convert all the Charecter [ASCII] to Binary & Store it
    for (int i = 0; i < n; i++)
        v[i] = get_binary((int)s[i]);

    // [Display to User]
    cout << "\nINPUT STRING\n \n[Character | ASCII | Binary Representation ]\n\n";
    for (int i = 0; i < n; i++)
    {
        cout << " " << s[i] << " -> " << (int)s[i] << " -> \t";
        for (int j = 0; j < 16; j++)
            cout << v[i][j] << " ";
        cout << endl;
    }
```

```
    vi sum(16, 0);
    vi check_sum(16, 0);

    // Step 2 : Add all the Data Bits
    for (int i = 0; i < n; i++)
        sum = add_binary(sum, v[i]);

    // [ Display Sum to User]
    cout << "\nSUM : \t \t";
    for (int i = 0; i < 16; i++)
    {
        cout << sum[i] << " ";
        // Since CheckSum is One's Complement of Sum
        if (sum[i] == 0)
            check_sum[i] = 1;
    }
    cout << endl;

    // Step 3 : Display CheckSum to User
    cout << "CHECKSUM : \t";
    for (int i = 0; i < 16; i++)
        cout << check_sum[i] << " ";

    return 0;
}

// Function to Convert the Number to Binary Form
vi get_binary(int n)
{
    vi bin(16, 0);
    int pos = 15;
    while (n)
    {
        bin[pos--] = n % 2;
        n /= 2;
    }
    return bin;
}

// 1's Complement Addition [Given there Binary Forms]
vi add_binary(vi num1, vi num2)
{
    int carry = 0, S;
    vi complement(16, 0);

    for (int i = 15; i >= 0; i--)
    {
        S = carry + num1[i] + num2[i];
        // CASE 1 : {0, 0, 0}
```

```
        if (S == 0)
            complement[i] = 0, carry = 0;
        // CASE 2 : {1, 0, 0}, {0, 1, 0}, {0, 0, 1}
        else if (S == 1)
            complement[i] = 1, carry = 0;
        // CASE 3 : {1, 1, 0}, {1, 0, 1}, {0, 1, 1}
        else if (S == 2)
            complement[i] = 0, carry = 1;
        // CASE 4 : {1, 1, 1}
        else
            complement[i] = 1, carry = 1;
    }

    //  If Carry is Set {Need to Add 1 to Complement}
    if (carry)
    {
        vi one(16, 0);
        one[15] = 1;
        complement = add_binary(complement, one);
    }

    return complement;
}
```

Test Case

```
~~~~~~~~~~~~~~~~~~~~~~~~ Sender Side ~~~~~~~~~~~~~~~~~~~~~~~~

Enter Message to be Encoded (string) : Forouzan

INPUT STRING  Binary Data Bits of Message

[Character | ASCII | Binary Representation ]

 F -> 70 ->      0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0
 o -> 111 ->     0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1
 r -> 114 ->     0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 0
 o -> 111 ->     0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1
 u -> 117 ->     0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 1
 z -> 122 ->     0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 0
 a -> 97 ->      0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1
 n -> 110 ->     0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 0

SUM :            0 0 0 0 0 0 1 1 0 1 0 1 0 1 0 0
CHECKSUM :       1 1 1 1 1 1 0 0 1 0 1 0 1 0 1 1
PS C:\Users\Admin\Desktop\CN_L4\CHECKSUM>
```

Code Implementation [Receiver Side]

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef vector<int> vi;
vi pow2(16, 0);

// Function to Precompute the Power's of 2
void pre();

// Function to Convert Binary to Decimal
int to_decimal(vi bin);

// 1's Complement Addition [Given there Binary Forms]
vi add_binary(vi num1, vi num2);

int main()
{
    // Power's of 2
    pre();

    cout << "\n~~~~~~~~~~~~~~~~~~~~~ Receiver Side ~~~~~~~~~~~~~~~~~~~~~\n\n";

    cout << "Enter the Number of Data Segments : ";
    int k, bit;
    cin >> k;

    // ds + checksum
    vi v[k + 1];

    for (int i = 0; i < k; i++)
    {
        cout << "Data Segment " << i + 1 << " : ";
        for (int j = 0; j < 16; j++)
        {
            cin >> bit;
            v[i].push_back(bit);
        }
    }

    cout << "Enter CHECKSUM : ";
    for (int j = 0; j < 16; j++)
    {
        cin >> bit;
        v[k].push_back(bit);
    }

    vi sum(16, 0);
    vi complement(16, 0);
```

```
    // Step 2 : Add all the Data Bits [Data Bits + Check Sum]
    for (int i = 0; i <= k; i++)
        sum = add_binary(sum, v[i]);

    bool error = false;

    // [Display the Final Sum]
    // Step 3 : If the Result is all 1's, ACCEPT; ELSE REJECT
    cout << "\nSUM : \t \t";
    for (int i = 0; i < 16; i++)
    {
        cout << sum[i] << " ";
        if (sum[i] == 0)
            complement[i] = 1, error = true;
        else
            complement[i] = 0;
    }

    cout << "\n\nCOMPLEMENT : \t";
    for (int i = 0; i < 16; i++)
    {
        cout << complement[i] << " ";
    }

    string msg;
    if (error)
    {
        cout << "\n\nError Detected!";
    }
    else
    {
        for (int i = 0; i < k; i++)
            msg += (char)(to_decimal(v[i]));

        cout << "\n\nError Not Detected!\n";
        cout << "Original Message Transmitted : " << msg << endl;
    }

    return 0;
}

// Function to Precompute the Power's of 2
void pre()
{
    pow2[0] = 1;
    for (int i = 1; i < 16; i++)
        pow2[i] = pow2[i - 1] * 2;
}

// Function to Convert Binary to Decimal
```

```
int to_decimal(vi bin)
{
    int num = 0;
    for (int i = 15; i >= 0; i--)
        num += bin[i] * pow2[15 - i];
    return num;
}

// 1's Complement Addition [Given there Binary Forms]
vi add_binary(vi num1, vi num2)
{
    int carry = 0, S;
    vi complement(16, 0);

    for (int i = 15; i >= 0; i--)
    {
        S = carry + num1[i] + num2[i];
        // CASE 1 : {0, 0, 0}
        if (S == 0)
            complement[i] = 0, carry = 0;
        // CASE 2 : {1, 0, 0}, {0, 1, 0}, {0, 0, 1}
        else if (S == 1)
            complement[i] = 1, carry = 0;
        // CASE 3 : {1, 1, 0}, {1, 0, 1}, {0, 1, 1}
        else if (S == 2)
            complement[i] = 0, carry = 1;
        // CASE 4 : {1, 1, 1}
        else
            complement[i] = 1, carry = 1;
    }

    //  If Carry is Set {Need to Add 1 to Complement}
    if (carry)
    {
        vi one(16, 0);
        one[15] = 1;
        complement = add_binary(complement, one);
    }

    return complement;
}
```

Test Case

```
~~~~~~~~~~~~~~~~~~~~~~ Receiver Side ~~~~~~~~~~~~~~~~~~~~~~

Enter the Number of Data Segments : 8
Data Segment 1 : 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0
Data Segment 2 : 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1
Data Segment 3 : 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 0
Data Segment 4 : 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1
Data Segment 5 : 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 1
Data Segment 6 : 0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 0
Data Segment 7 : 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1
Data Segment 8 : 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 0
Enter CHECKSUM : 1 1 1 1 1 1 0 0 1 0 1 0 1 0 1 1

SUM :           1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1   <- All 1's

COMPLEMENT :    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Error Not Detected!
Original Message Transmitted : Forouzan
PS C:\Users\Admin\Desktop\CN_L4\CHECKSUM>
```

Made Some Changes in Data Bits

```
~~~~~~~~~~~~~~~~~~~~~~ Receiver Side ~~~~~~~~~~~~~~~~~~~~~~

Enter the Number of Data Segments : 8
Data Segment 1 : 0 0 0 0 1 0 0 0 0 1 0 0 0 1 1 0
Data Segment 2 : 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1
Data Segment 3 : 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 0
Data Segment 4 : 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1
Data Segment 5 : 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 1
Data Segment 6 : 0 1 0 0 0 0 0 0 0 1 1 1 1 0 1 0
Data Segment 7 : 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 1
Data Segment 8 : 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 0
Enter CHECKSUM : 1 1 1 1 1 1 0 0 1 0 1 0 1 0 1 1

SUM :           0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0

COMPLEMENT :    1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1   Not all 1's

Error Detected!
PS C:\Users\Admin\Desktop\CN_L4\CHECKSUM>
```

SUBMITTED BY:

U19CS012 [BHAGYA VINOD RANA]