EAST WEST UNIVERSITY

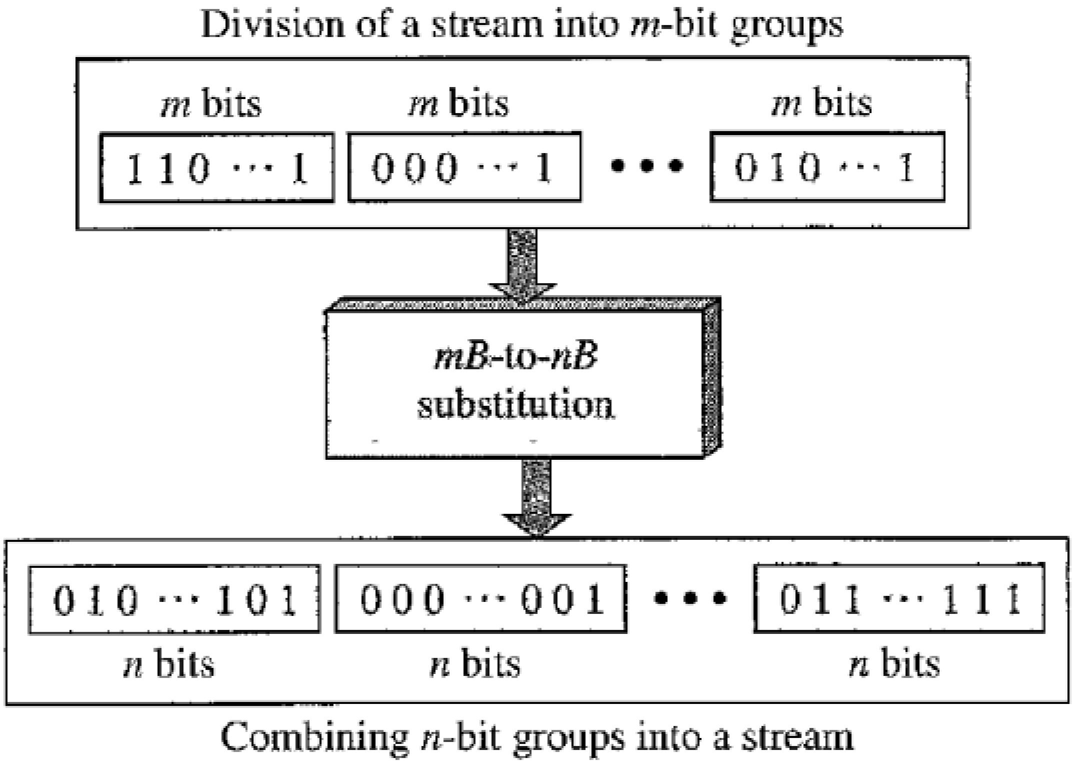
Department of CSE
Lab Report

Course Code and Name: CSE350 - Data communication	
Lab: Block Coding Concept	
Name of Student: Intesar Islam Khan ID: 2019-1-60-043	**Course Instructor information:** **Md. Mahir Ashhab** Lecturer Department of Computer Science & Engineering
Section: 02	**Date of Submission:** 28/11/2022

Source code: https://github.com/IntesarEWU/CSE350-Data-communication-LAB/tree/main/Lab5

Introduction:

4B/5B encoding is a type of 'Block coding'. This processes groups of bits rather than outputting a signal for each individual bit (as in Manchester encoding). A group of 4 bits is encoded so that an extra 5th bit is added. Since the input data is taken 4-bits at a time, there are 2^4, or 16 different bit patterns.



4B/5B mapping codes:

Data Sequence	Encoded Sequence
0000	11110
0001	01001
0010	10100
0011	10101
0100	01010
0101	01011
0110	01110
0111	01111
1000	10010
1001	10011
1010	10110
1011	10111
1100	11010
1101	11011
1110	11100

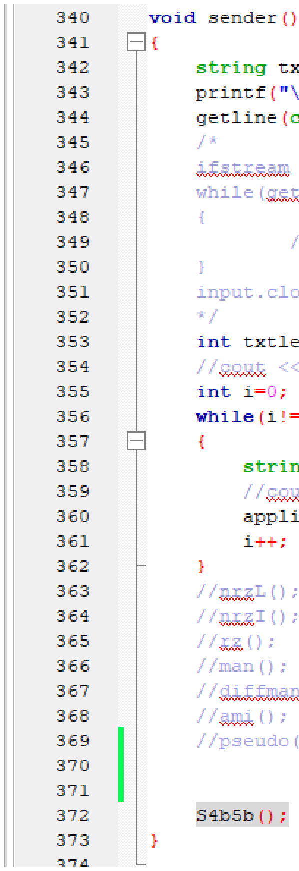
Lab Task:

Creating a sender receiver environment to show how block coding actually works.

Implementation:

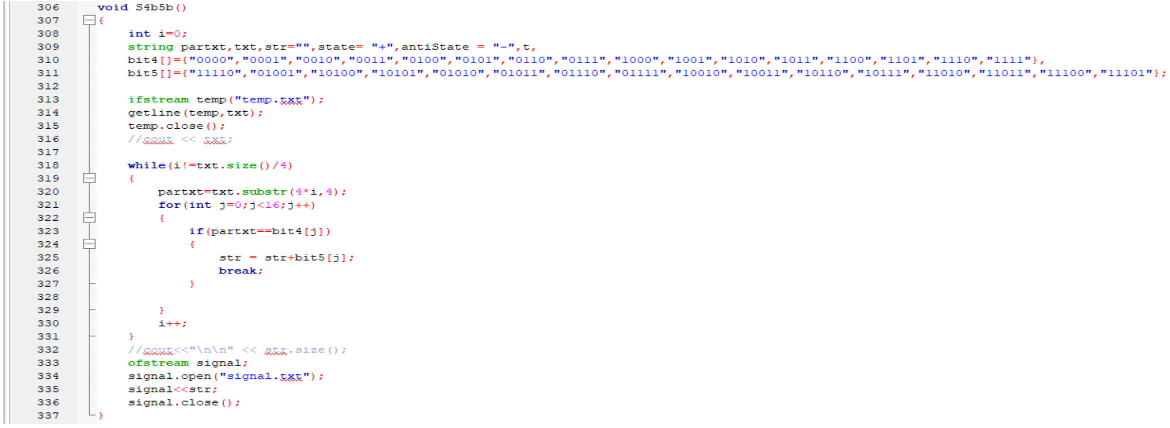
sender.cpp: to create the Block coding environment I create one function which will divide binary string in to 4-bit binary and convert that into a 5-bit binary and then combine them together.

sender (function): reads text file or takes user input and send the string to application layer and complete the OSI model and then start the blocking coding part.



```
void sender()
{
    string txt;
    printf("\nSender  : ");
    getline(cin,txt);
    /*
    ifstream input("input.txt");
    while(getline(input,txt))
    {
            //cout << str << '\n';
    }
    input.close()
    */
    int txtlen = txt.size();
    //cout << txtlen <<'\n';
    int i=0;
    while(i!=(txtlen/125)+1)
    {
        string str = txt.substr(125*i,125);
        //cout << str << '\n';
        applicationLayer(str);
        i++;
    }
    //nrzL();
    //nrzI();
    //rz();
    //man();
    //diffman();
    //ami();
    //pseudo();


    S4b5b();
}
```

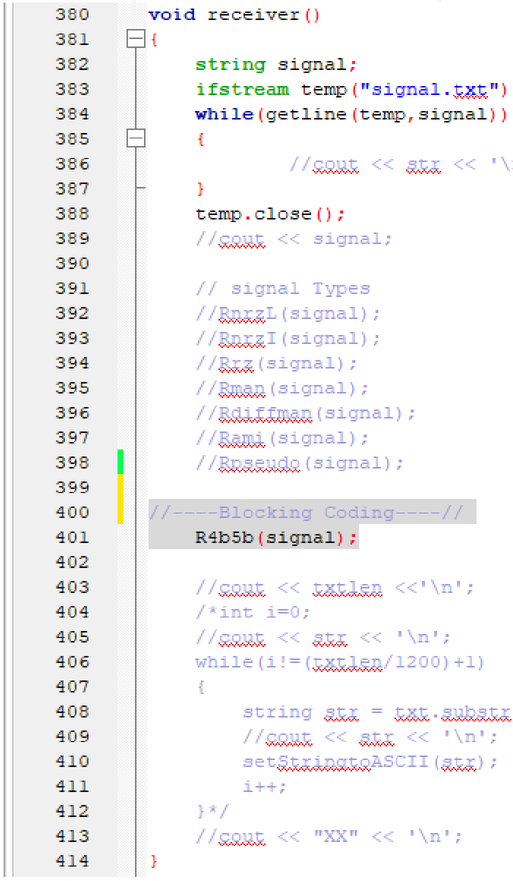
4B/5B function: It reads the full "temp.txt" file which contains binary of sender text and divide the binary string to 4 bits and then convert the 4-bit binary to 5-bit binary correspondingly



```
void S4b5b()
{
    int i=0;
    string partxt,txt,str="",state= "+",antiState = "-",t,
    bit4[]={"0000","0001","0010","0011","0100","0101","0110","0111","1000","1001","1010","1011","1100","1101","1110","1111"},
    bit5[]={"11110","01001","10100","10101","01010","01011","01110","01111","10010","10011","10110","10111","11010","11011","11100","11101"};

    ifstream temp("temp.txt");
    getline(temp,txt);
    temp.close();
    //cout << txt;

    while(i!=txt.size()/4)
    {
        partxt=txt.substr(4*i,4);
        for(int j=0;j<16;j++)
        {
            if(partxt==bit4[j])
            {
                str = str+bit5[j];
                break;
            }

        }
        i++;
    }
    //cout<<"\n\n" << str.size();
    ofstream signal;
    signal.open("signal.txt");
    signal<<str;
    signal.close();
}
```

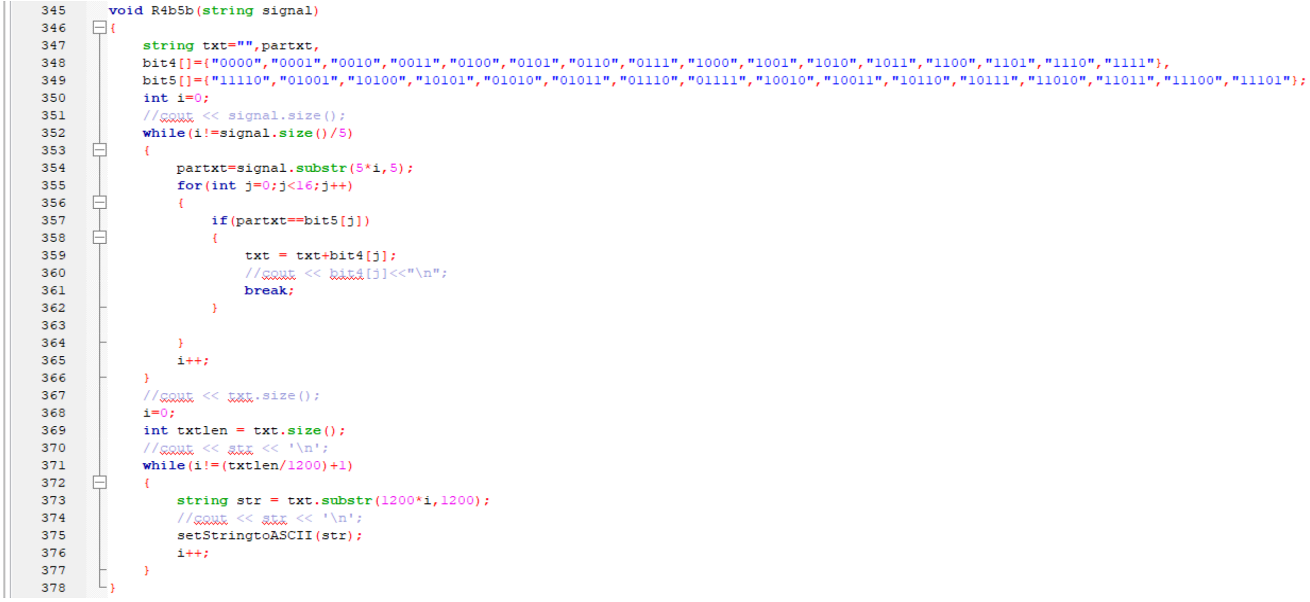
receiver.cpp: to create the Block coding environment I create one function which will divide binary string in to 5-bit binary and convert that into a 4-bit binary and then combine them together.

Receiver function: it reads the singnal.txt file and send the signal string to the 5B/4B function.



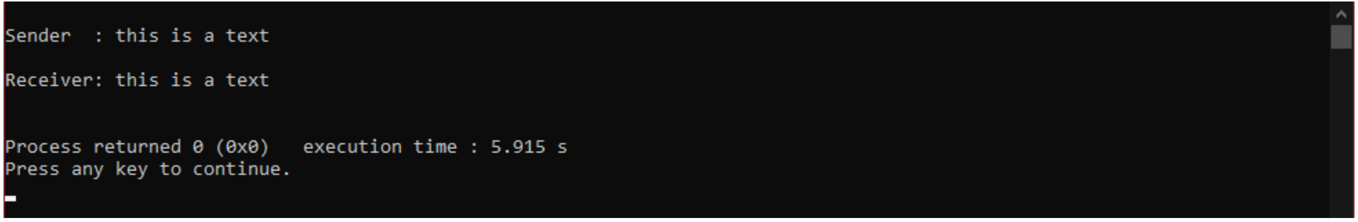
```
void receiver()
{
    string signal;
    ifstream temp("signal.txt");
    while(getline(temp,signal))
    {
            //cout << str << '\n';
    }
    temp.close();
    //cout << signal;

    // signal Types
    //RnrzL(signal);
    //RnrzI(signal);
    //Rrz(signal);
    //Rman(signal);
    //Rdiffman(signal);
    //Rami(signal);
    //Rpseudo(signal);

//----Blocking Coding----//
    R4b5b(signal);

    //cout << txtlen <<'\n';
    /*int i=0;
    //cout << str << '\n';
    while(i!=(txtlen/1200)+1)
    {
        string str = txt.substr(1200*i,1200);
        //cout << str << '\n';
        setStringtoASCII(str);
        i++;
    }*/
    //cout << "XX" << '\n';
}
```

4B/5B function: in receiver side the signal string will divide in 5-bit binary string and the converted to 4-bit binary string correspondingly and combine together.



```
void R4b5b(string signal)
{
    string txt="",partxt,
    bit4[]={"0000","0001","0010","0011","0100","0101","0110","0111","1000","1001","1010","1011","1100","1101","1110","1111"},
    bit5[]={"11110","01001","10100","10101","01010","01011","01110","01111","10010","10011","10110","10111","11010","11011","11100","11101"};
    int i=0;
    //cout << signal.size();
    while(i!=signal.size()/5)
    {
        partxt=signal.substr(5*i,5);
        for(int j=0;j<16;j++)
        {
            if(partxt==bit5[j])
            {
                txt = txt+bit4[j];
                //cout << bit4[j]<<"\n";
                break;
            }

        }
        i++;
    }
    //cout << txt.size();
    i=0;
    int txtlen = txt.size();
    //cout << str << '\n';
    while(i!=(txtlen/1200)+1)
    {
        string str = txt.substr(1200*i,1200);
        //cout << str << '\n';
        setStringtoASCII(str);
        i++;
    }
}
```

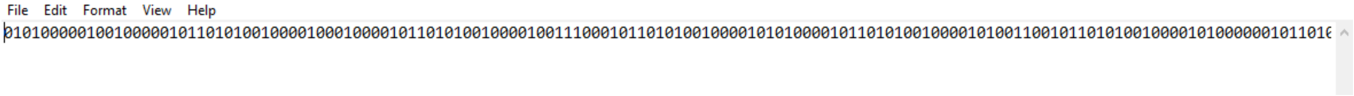
Outputs:



```
Sender  : this is a text

Receiver: this is a text


Process returned 0 (0x0)   execution time : 5.915 s
Press any key to continue.
```

temp.txt:

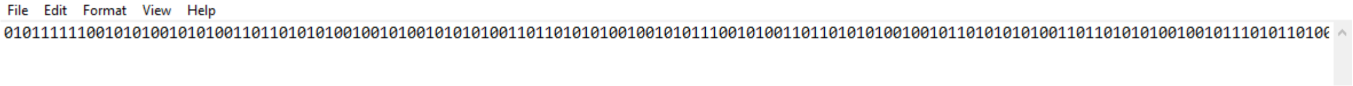
this file contains the binary string which size is now 312



```
0101000001001000001011101010010000100010000101101010010000100111000101101010010000101010000101101010010000101001100101101010010000101000000101101
```

signal.txt:

this file contains the 4b/5b signal string which size is now `390 = ((312 / 4) * 5)`



```
0101111110010101001010100110110101010010010100101010100110110101010010010101110010100110110101010010010110101010100110110101010010010111010110100
```

Discursion:

This block coding part of this application was not really that hard. I just had to create 2 string arrays for both sender and receiver side and the compare the divided parts of the sting with these arrays and just combine the corresponding binary strings and that’s it.