EAST WEST UNIVERSITY

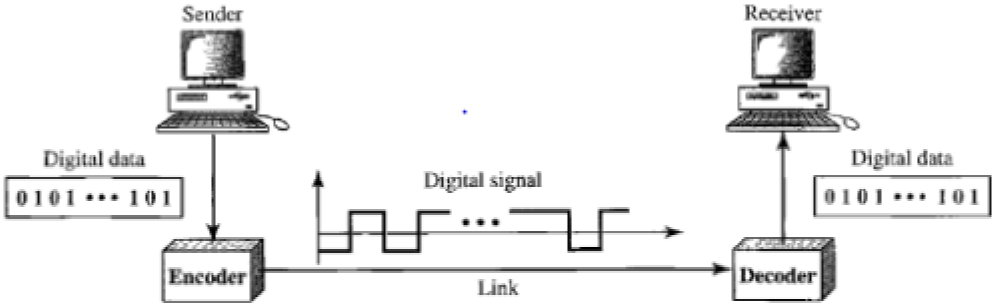
## Department of CSE

## Lab Report

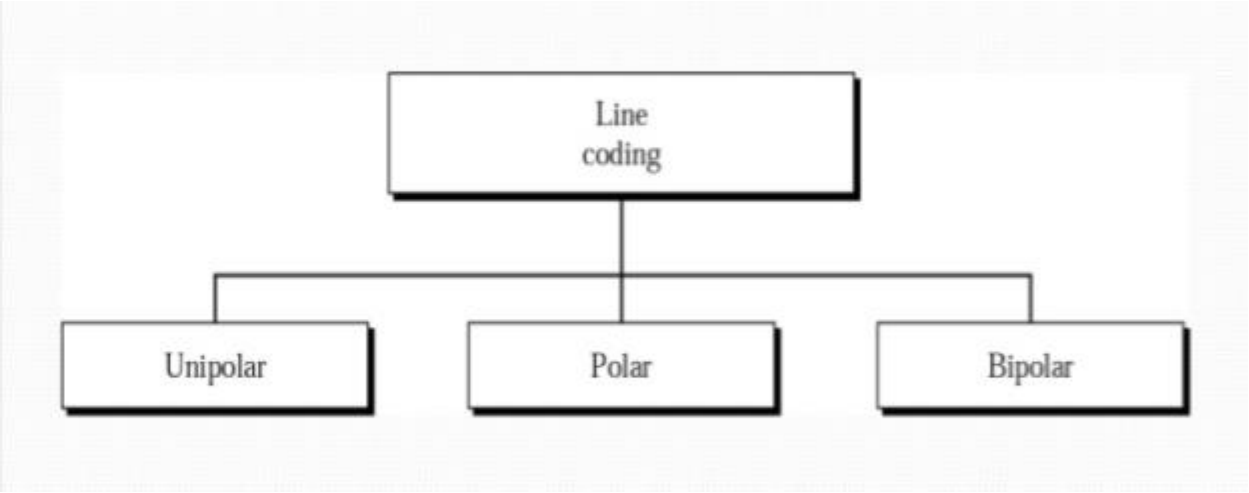
| | |
|---|---|
| **Course Code and Name:**<br>CSE350 – Data communication | |
| **Lab:** | |
| Name of Student:<br>Intesar Islam Khan<br>ID: 2019-1-60-043 | Course Instructor information:<br>Md. Mahir Ashhab<br>Lecturer<br>Department of Computer Science & Engineering |
| **Section:** 02 | **Date of Submission:**<br>14/11/2022 |

## Introduction:

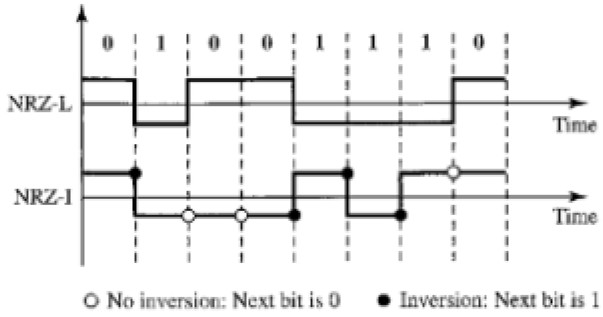
Line coding is the process of converting digital data to signal data. This data can be either digital or analog. The conversion involves three techniques: line coding, block coding, and scrambling.

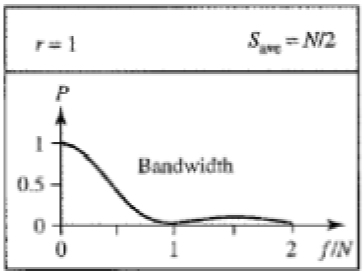


In line coding, there are several schemes in each category.

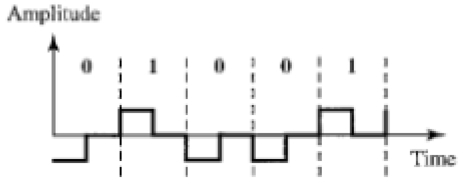


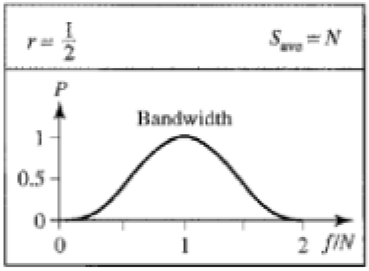
**Non-Return to Zero(NRZ):** In polar NRZ encodings, there are two levels of voltage amplitude and two versions of polar NRZ: NRZ-L and NRZ-I. For NRZ-L On the sender side if the binary value is zero then the signal will be + and if the binary value is one then the signal will be minus. For NRZ-I if the binary value is 0, a signal will be the same as previous signal, and if one it will invert.



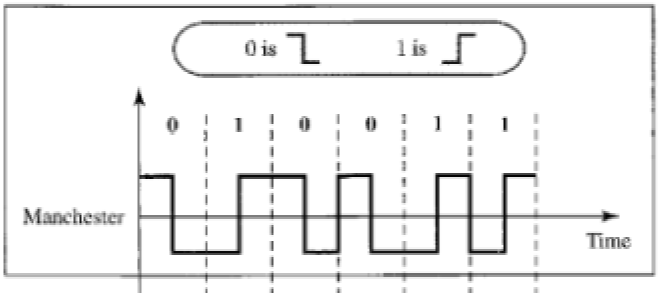


**Return to Zero (RZ):** In sender side if binary value is zero we have to print minus zero in signal and when binary value is one that time print +0.

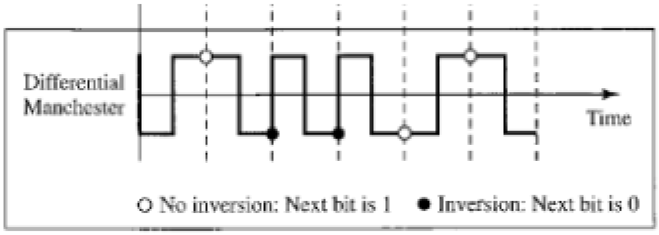




**Manchester:** In the sender side if the binary value is 0 print '+-' and if the value is 1 print '-+'.



**Differential Manchester**: In the sender side if binary value is 1, it will be continuing and 0 then print invert.



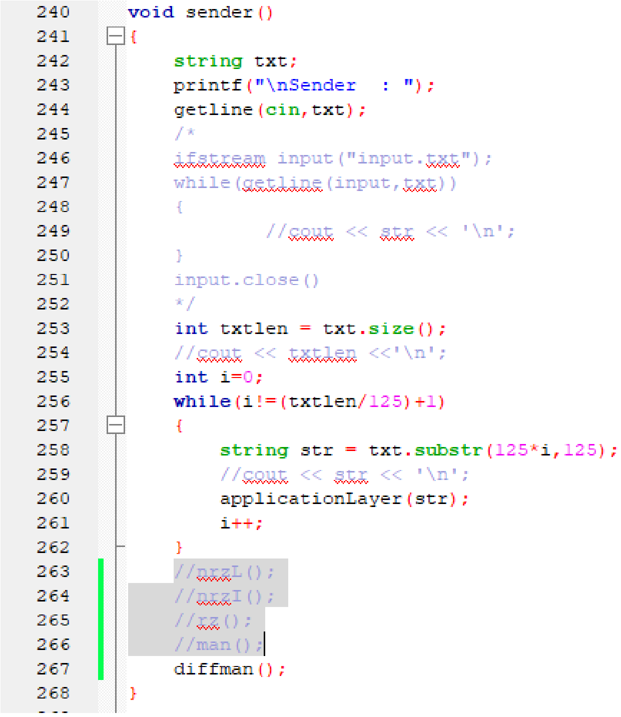
## Lab Task:

Creating a sender receiver environment to show how OSI model works and print every 125-character segment from receiver side

## Implementation:

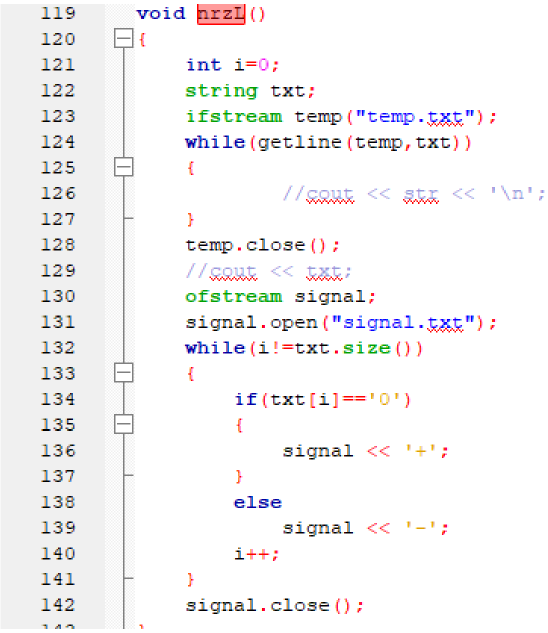
I created 3 separated c++ files 1. main.cpp, 2. sender.cpp, 3. seceiver.cpp and also 4 text files 1. input.txt, 2. temp.txt, 3. output.txt, **4.signal.txt**

**sender.cpp:** to create the OSI environment I create 7 more function which works as layers.
**sender (function):** reads text file or takes user input and send the string to application layer.



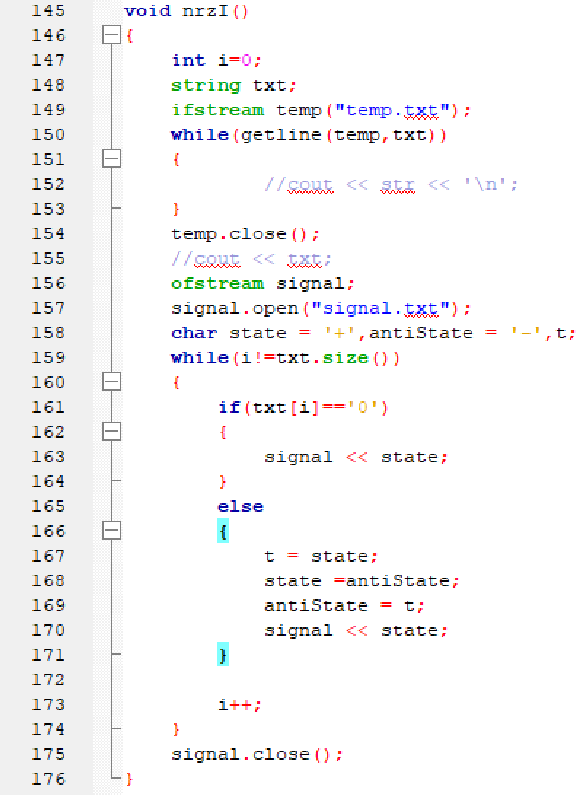
```cpp
void sender()
{
    string txt;
    printf("\nSender  : ");
    getline(cin,txt);
    /*
    ifstream input("input.txt");
    while(getline(input,txt))
    {
            //cout << str << '\n';
    }
    input.close()
    */
    int txtlen = txt.size();
    //cout << txtlen <<'\n';
    int i=0;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

NRZ-L:



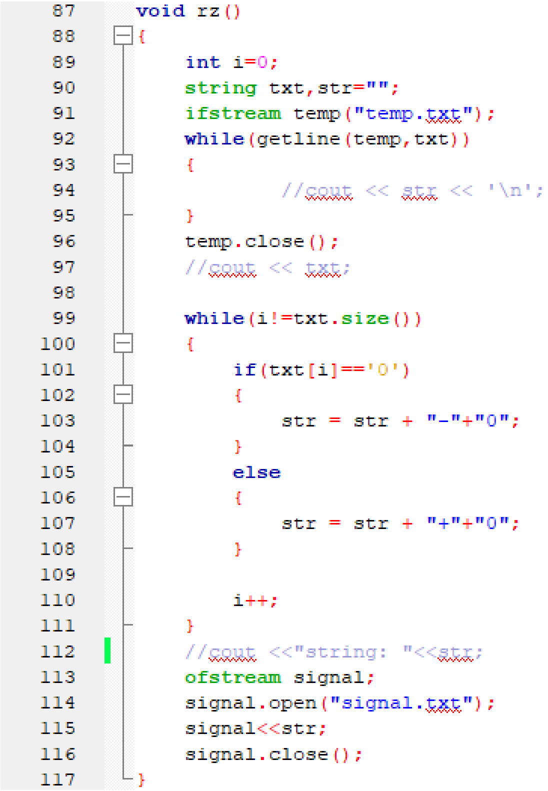
```cpp
void nrzL()
{
    int i=0;
    string txt;
    ifstream temp("temp.txt");
    while(getline(temp,txt))
    {
            //cout << str << '\n';
    }
    temp.close();
    //cout << txt;
    ofstream signal;
    signal.open("signal.txt");
    while(i!=txt.size())
    {
        if(txt[i]=='0')
        {
            signal << '+';
        }
        else
            signal << '-';
        i++;
    }
    signal.close();
```

NRZ-I:



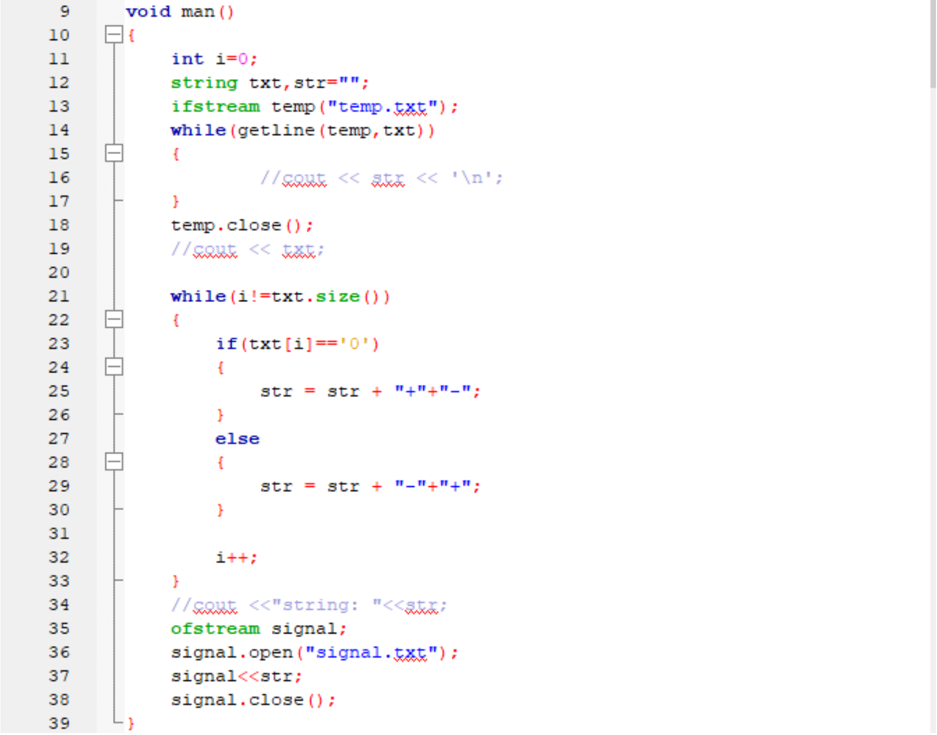
```
void nrzI()
{
    int i=0;
    string txt;
    ifstream temp("temp.txt");
    while(getline(temp,txt))
    {
            //cout << str << '\n';
    }
    temp.close();
    //cout << txt;
    ofstream signal;
    signal.open("signal.txt");
    char state = '+',antiState = '-',t;
    while(i!=txt.size())
    {
        if(txt[i]=='0')
        {
            signal << state;
        }
        else
        {
            t = state;
            state =antiState;
            antiState = t;
            signal << state;
        }

        i++;
    }
    signal.close();
}
```

RZ:



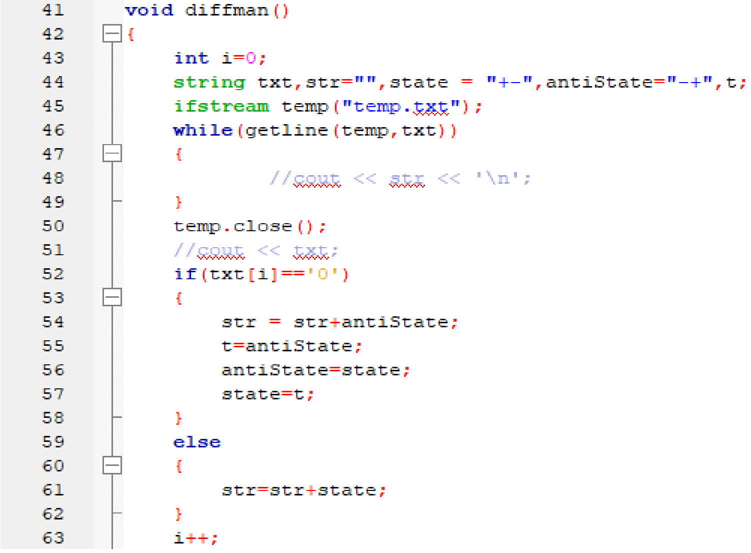
```
void rz()
{
    int i=0;
    string txt,str="";
    ifstream temp("temp.txt");
    while(getline(temp,txt))
    {
            //cout << str << '\n';
    }
    temp.close();
    //cout << txt;

    while(i!=txt.size())
    {
        if(txt[i]=='0')
        {
            str = str + "-"+"0";
        }
        else
        {
            str = str + "+"+"0";
        }

        i++;
    }
    //cout <<"string: "<<str;
    ofstream signal;
    signal.open("signal.txt");
    signal<<str;
    signal.close();
}
```

Manchester:

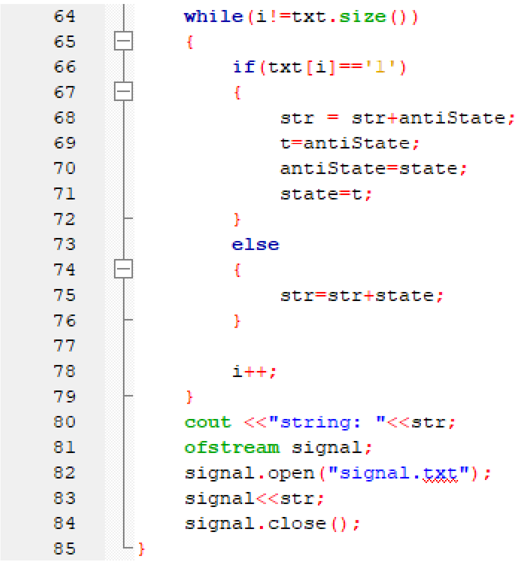


```
void man()
{
    int i=0;
    string txt,str="";
    ifstream temp("temp.txt");
    while(getline(temp,txt))
    {
            //cout << str << '\n';
    }
    temp.close();
    //cout << txt;

    while(i!=txt.size())
    {
        if(txt[i]=='0')
        {
            str = str + "+"+"-";
        }
        else
        {
            str = str + "-"+"+";
        }

        i++;
    }
    //cout <<"string: "<<str;
    ofstream signal;
    signal.open("signal.txt");
    signal<<str;
    signal.close();
}
```

**Differential Manchester**:

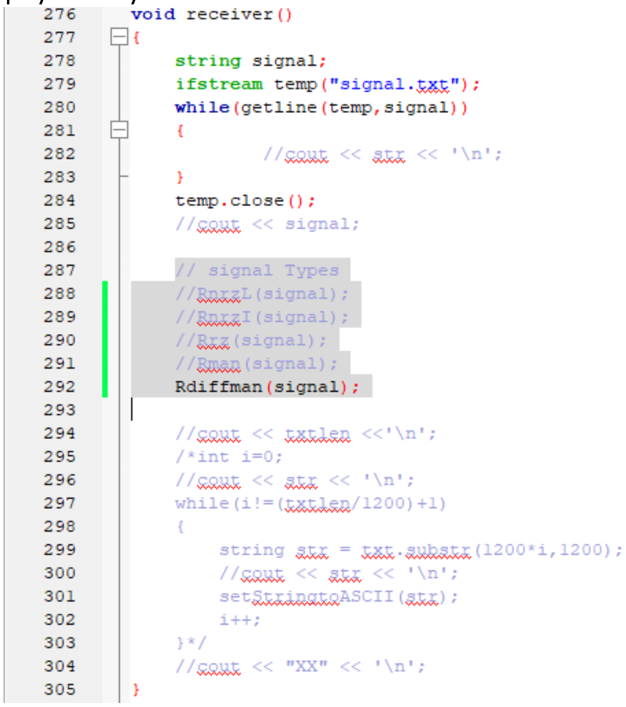


```
void diffman()
{
    int i=0;
    string txt,str="",state = "+-",antiState="-+",t;
    ifstream temp("temp.txt");
    while(getline(temp,txt))
    {
            //cout << str << '\n';
    }
    temp.close();
    //cout << txt;
    if(txt[i]=='0')
    {
        str = str+antiState;
        t=antiState;
        antiState=state;
        state=t;
    }
    else
    {
        str=str+state;
    }
    i++;
```



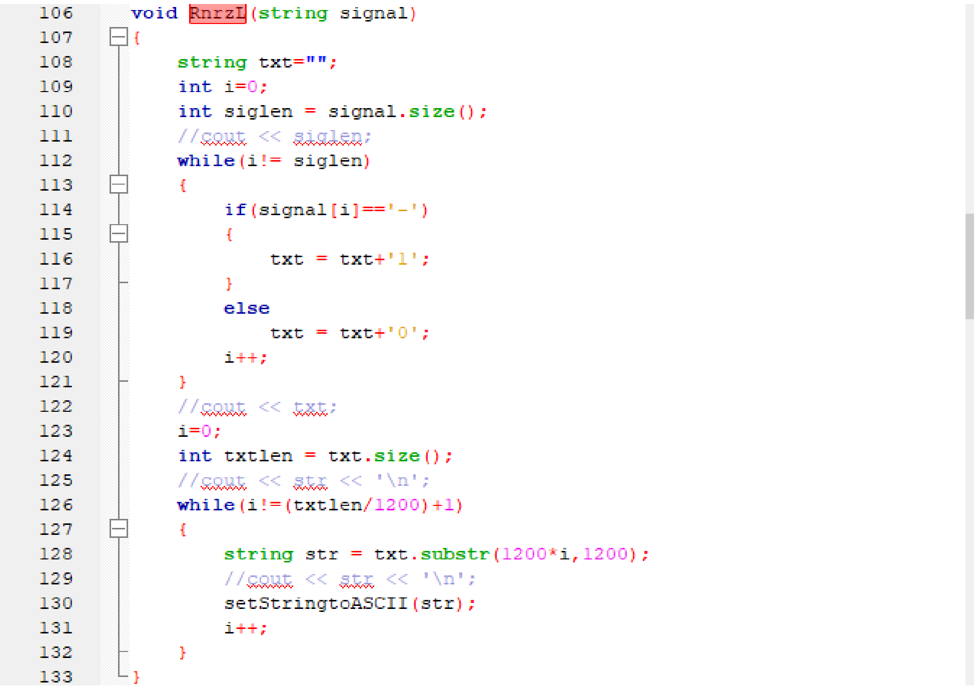
```
    while(i!=txt.size())
    {
        if(txt[i]=='1')
        {
            str = str+antiState;
            t=antiState;
            antiState=state;
            state=t;
        }
        else
        {
            str=str+state;
        }

        i++;
    }
    cout <<"string: "<<str;
    ofstream signal;
    signal.open("signal.txt");
    signal<<str;
    signal.close();
}
```

**receiver.cpp:** to create the OSI environment I create 7 more function which works as layers. **receiver (function):** reads text file (temp.txt) and convert the binary string to char string and send the string to physical layer.



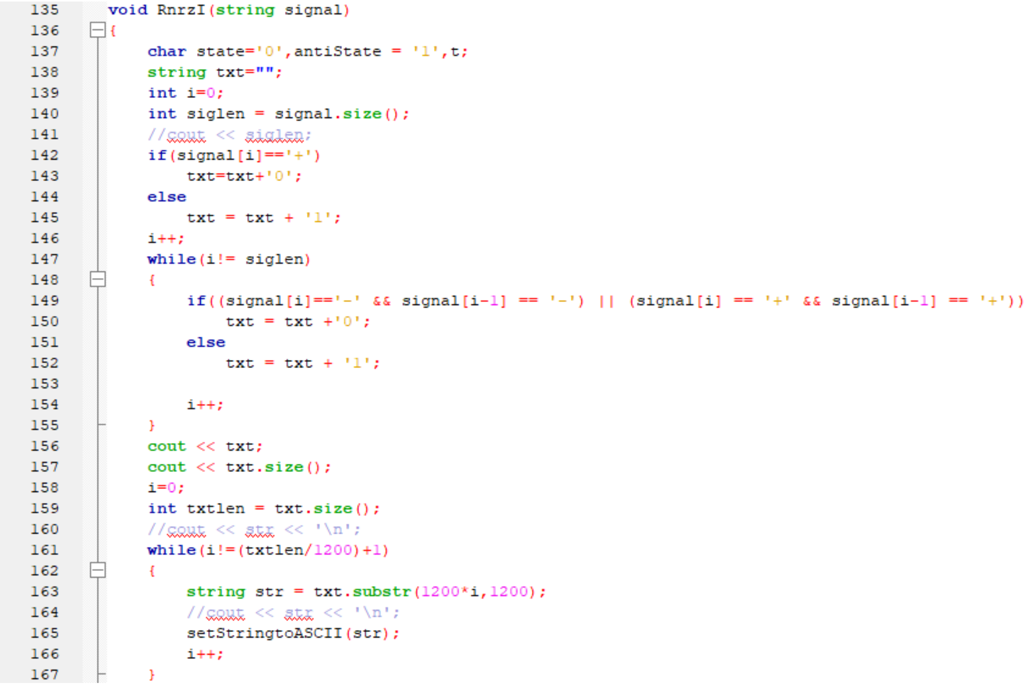
```
void receiver()
{
    string signal;
    ifstream temp("signal.txt");
    while(getline(temp,signal))
    {
            //cout << str << '\n';
    }
    temp.close();
    //cout << signal;

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

    //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';
}
```

NRZ-L:



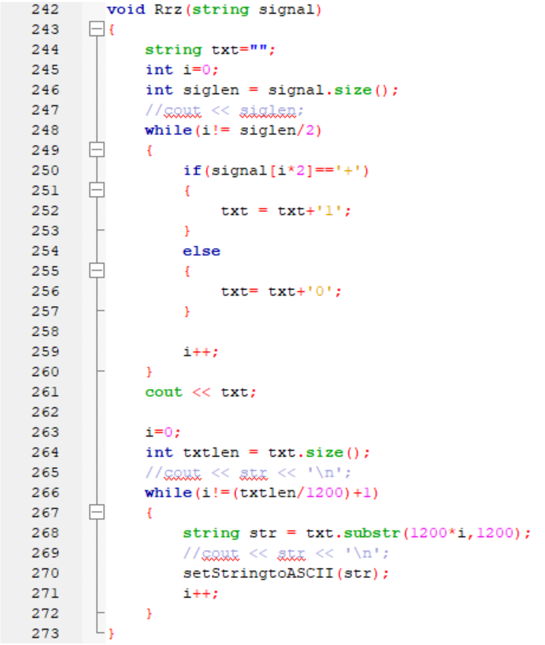
```
void RnrzL(string signal)
{
    string txt="";
    int i=0;
    int siglen = signal.size();
    //cout << siglen;
    while(i!= siglen)
    {
        if(signal[i]=='-')
        {
            txt = txt+'1';
        }
        else
            txt = txt+'0';
        i++;
    }
    //cout << txt;
    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++;
    }
}
```

NRZ-I:



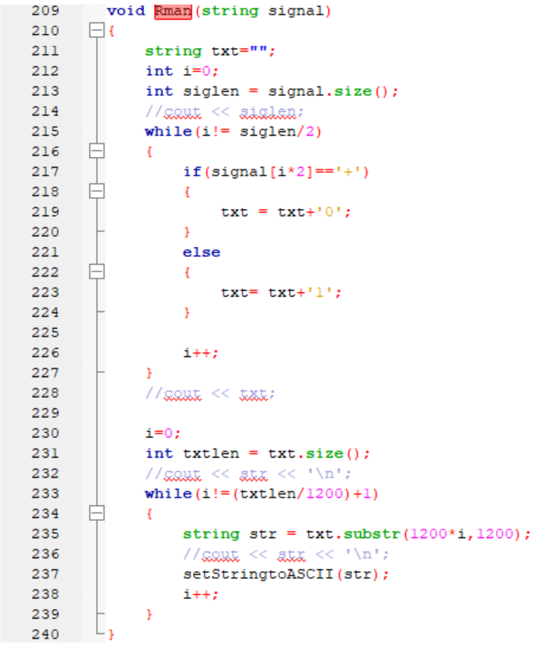
```
void RnrzI(string signal)
{
    char state='0',antiState = '1',t;
    string txt="";
    int i=0;
    int siglen = signal.size();
    //cout << siglen;
    if(signal[i]=='+')
        txt=txt+'0';
    else
        txt = txt + '1';
    i++;
    while(i!= siglen)
    {
        if((signal[i]=='-' && signal[i-1] == '-') || (signal[i] == '+' && signal[i-1] == '+'))
            txt = txt +'0';
        else
            txt = txt + '1';

        i++;
    }
    cout << txt;
    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++;
    }
```

RZ:



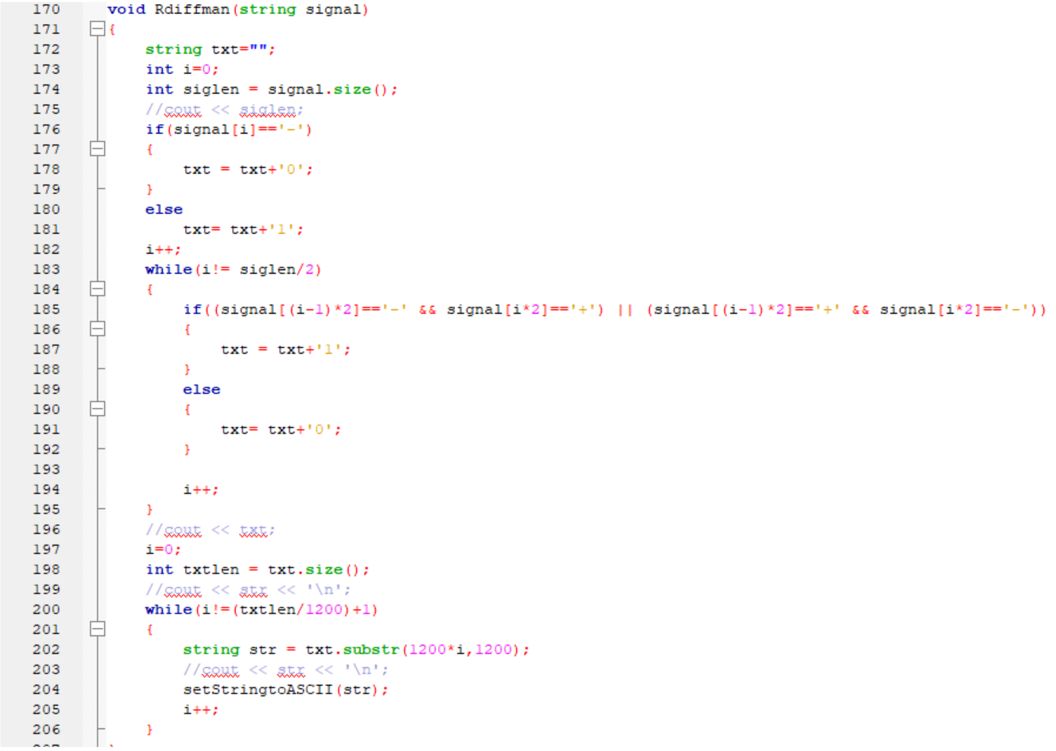
```
void Rrz(string signal)
{
    string txt="";
    int i=0;
    int siglen = signal.size();
    //cout << siglen;
    while(i!= siglen/2)
    {
        if(signal[i*2]=='+')
        {
            txt = txt+'1';
        }
        else
        {
            txt= txt+'0';
        }

        i++;
    }
    cout << txt;

    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++;
    }
}
```

Manchester:



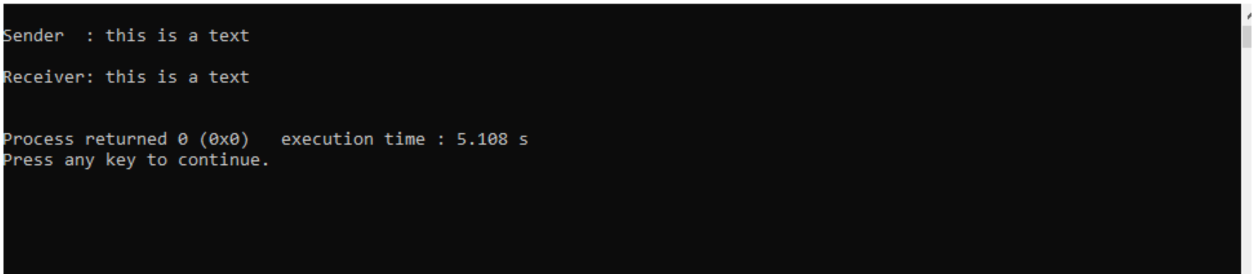
```
void Rman(string signal)
{
    string txt="";
    int i=0;
    int siglen = signal.size();
    //cout << siglen;
    while(i!= siglen/2)
    {
        if(signal[i*2]=='+')
        {
            txt = txt+'0';
        }
        else
        {
            txt= txt+'1';
        }

        i++;
    }
    //cout << txt;

    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++;
    }
}
```

## Differential Manchester:



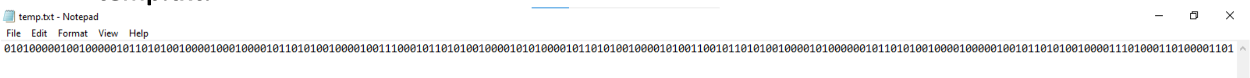
```
void Rdiffman(string signal)
{
    string txt="";
    int i=0;
    int siglen = signal.size();
    //cout << siglen;
    if(signal[i]=='-')
    {
        txt = txt+'0';
    }
    else
        txt= txt+'1';
    i++;
    while(i!= siglen/2)
    {
        if((signal[(i-1)*2]=='-' && signal[i*2]=='+') || (signal[(i-1)*2]=='+' && signal[i*2]=='-'))
        {
            txt = txt+'1';
        }
        else
        {
            txt= txt+'0';
        }

        i++;
    }
    //cout << txt;
    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: Sample output for a single function.(RZ)



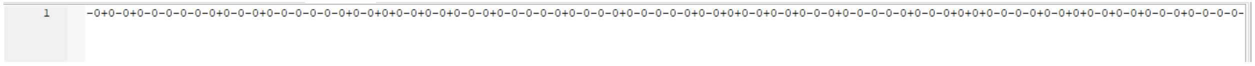
```
Sender  : this is a text

Receiver: this is a text


Process returned 0 (0x0)   execution time : 5.108 s
Press any key to continue.
```

### temp.txt:



```
0101000001001000001011010100100001000100001011010100100001001110001011010100100001010100001011010100100001010011001011010100100001010000001011010100100001000001001011010100100001110100011010000110 1
```

### output.txt:



```
-0+0-0+0-0-0-0-0-0+0-0-0+0-0-0-0-0-0+0-0+0+0-0+0-0+0-0-0+0-0-0-0-0+0-0-0-0+0-0-0-0-0+0-0+0+0-0+0-0+0-0-0+0-0-0-0-0+0-0-0+0+0+0-0-0-0+0-0+0+0-0+0-0+0-0-0+0-0-0-0-
```

## Discussion:

While I was creating this application, I did not face any problem, mainly because I remember the pseudocode for every function and applied as it is.