**Name:** Mitrajeet Golsangi

**Roll No:** 01

**PRN:** 12010484

**Division:** Ty-CS-B

**Batch:** B1

**TY. B. Tech.**

**CS3052: Computer Networks**

**Lab No: 4**

**Write a program for error detection and correction for 7/8 bits ASCII codes using Hamming Codes or CRC. Demonstrate the packets captured traces using Wireshark Packet Analyzer Tool for peer to peer mode.**

**Code:**

*#include* <algorithm>

*#include* <conio.h>

*#include* <iostream>

using namespace std;

string codeword;

void tocodeword(string t)

{

    int p1 = (((t[3] + t[2] + t[0]) - (3 \* '0'))) % 2; *// 3 5 7*

    int p2 = (((t[3] + t[1] + t[0]) - (3 \* '0'))) % 2; *// 3 6 7*

    int p4 = (((t[2] + t[1] + t[0]) - (3 \* '0'))) % 2; *// 5 6 7*

*// codeword.append(t[0]+t[1]+t[2]++t[3]+('0'+p2)+('0'+p1);*

    codeword.append(t.substr(0, 3));

    codeword.append(string(1, ('0' + p4)));

    codeword.append(string(1, t[3]));

    codeword.append(string(1, ('0' + p2)));

    codeword.append(string(1, ('0' + p1)));

}

void correctcode(string t)

{

    int p1 = (((t[6] + t[4] + t[2] + t[0]) - (4 \* '0')) % 2); *// 3 5 7*

    int p2 = (((t[5] + t[4] + t[1] + t[0]) - (4 \* '0')) % 2); *// 3 6 7*

    int p4 = (((t[3] + t[2] + t[1] + t[0]) - (4 \* '0')) % 2); *// 5 6 7*

*if* (p1 + p2 + p4 == 0)

    {

        cout << "No Error in the token\n";

*return*;

    }

    cout << "\nToken :" << t << endl;

    cout << "\nFaulty Index in token: ";

    reverse(t.begin(), t.end());

*// cout << p1 << " " << p2 << " " << " " << p4;*

    int f = p1 \* 1 + p2 \* 2 + p4 \* 4 - 1;

    cout << f + 1;

    t[f] == '1' ? t[f] = '0' : t[f] = '1';

    reverse(t.begin(), t.end());

    cout << "\nCorrected Token :" << t << endl;

}

int main()

{

    string data;

    srand(time(NULL));

    cout << "Enter the binary stream of data: ";

    cin >> data;

    int index = 0;

*while* (index < data.length())

    {

        tocodeword(data.substr(index, 4));

        index = index + 4;

    }

    cout << "\nCalculating... \n";

    cout << "The Generated/Tokenised Codeword is :" << codeword << "\n\n";

    int i = 0, j = 1;

*while* (i < data.length())

    {

        string tok = codeword.substr(i, 7);

        cout << "Token " << j++ << ":" << tok << "\n";

        i = i + 7;

    }

    cout << "Generating Random Error in the bitstream: ...\n\n";

    float poss;

    cout << "Enter the possibility (b/w 0 & 1) of error \n";

    cin >> poss;

    int ntk = codeword.length() / 7;

    const int ca = poss \* ntk;

*// cout << "Tokens Affected: "<< ca << endl;*

    int affected[ca];

*// int c = 0;*

*for* (int i = 0; i < ca; i++)

    {

        int r = ((rand() + i) % ca);

        int f = 1;

*for* (int j = i - 1; j >= 0; j--)

        {

*if* (affected[j] == r)

            {

                f = 0;

*// c++;*

                i--;

*break*;

            }

        }

*if* (f)

        {

            affected[i] = r;

*// cout << endl << " >" << i << " " << affected[i] << " <" << endl;*

        }

    }

*for* (int i = 0; i < ca; i++)

    {

        int x = rand() % 7;

        int ind = (affected[i] \* 7) + x;

        affected[i] = ind;

*if* (codeword[ind] == '1')

        {

            codeword[ind] = '0';

        }

*else*

        {

            codeword[ind] = '1';

        }

    }

    cout << "Error Codeword " << codeword << "\nIndices of Affected bits: ";

    int count = 0;

*for* (int i = 0; i < ca; i++)

    {

        count++;

        cout << affected[i] << " ";

    }

*if* (count == 0)

        cout << "No bit affected";

    cout << "\n\nCorrecting codeword .... \n";

    index = 0;

*while* (index < codeword.length())

    {

        cout << "\n\*\*\*Tokenwise Correction\*\n";

        correctcode(codeword.substr(index, 7));

        index = index + 7;

    }

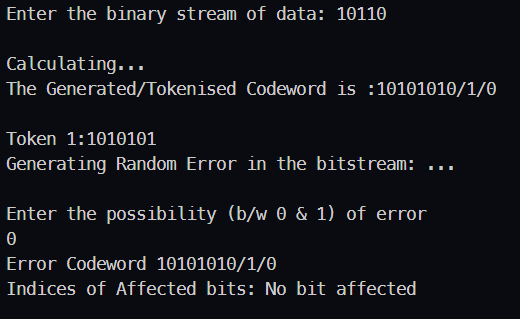
    cout << "\n\*\*\*\*End of tokenwise correction\*\*\n";

    cout << "\nCorrected Codeword is: " << codeword;

    getch();

}

**Output:**

****