DEPARTMENT OF COMPUTER ENGINEERING
Computer Network Lab

Semester	T.E. Semester V – Computer Engineering
Subject	Computer Network
Subject Professor In-charge	Prof. Amit K. Nerurkar
Assisting Teachers	Prof. Amit K. Nerurkar
Laboratory	M-313-A

Student Name	Deep Salunkhe
Roll Number	21102A0014
TE Division	A

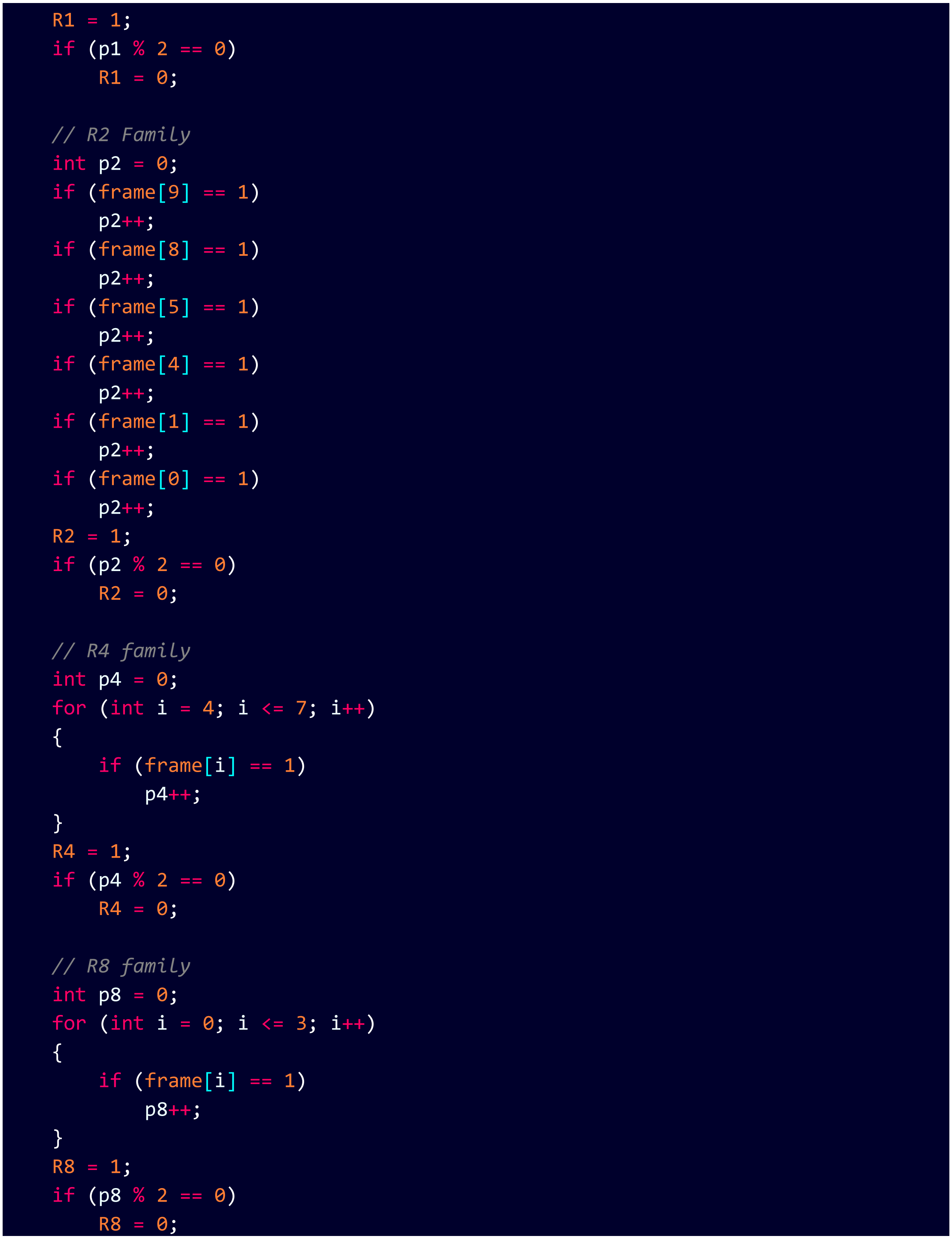
Title : Hamming Code

Theory:

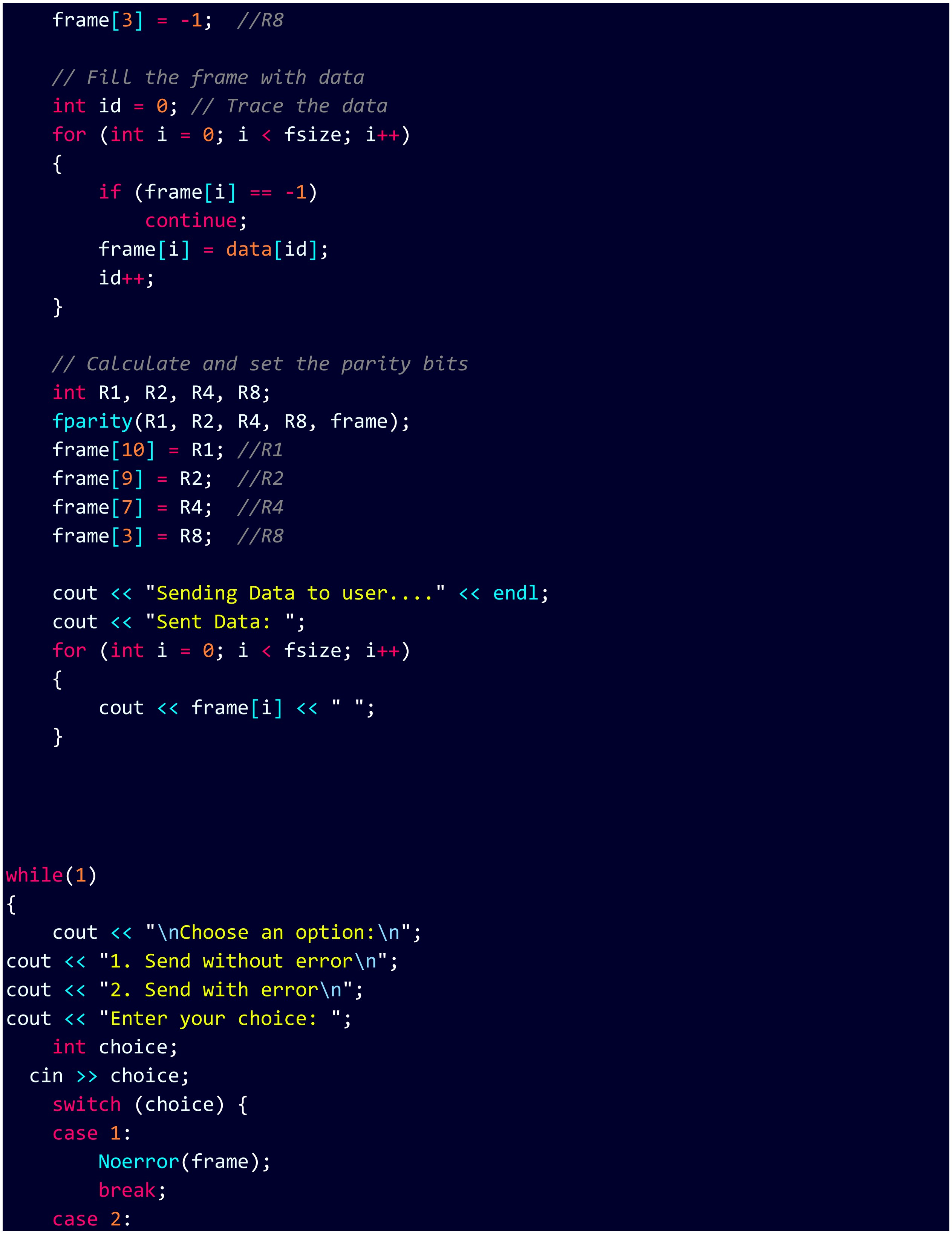
Hamming codes are designed to detect and correct single-bit errors. The receiver checks the received code word for errors using the parity bits. If an error is detected, the receiver can pinpoint the erroneous bit and correct it. If multiple errors occur, Hamming codes may not be able to correct them.

Implementation:

```
#include<iostream>
#include<cmath>
#include<vector>
#include<cstdlib>
#include<ctime>

using namespace std;

// Function to find the number of redundant bits required (r)
void findr(int &r)
{
    for (int i = 0; i < 7; i++)
    {
        if (pow(2, i) >= 7 + i + 1)
        {
            r = i;
            break;
        }
    }
}

// Function to calculate the parity bits (R1, R2, R4, R8)
void fparity(int &R1, int &R2, int &R4, int &R8, vector<int> frame)
{
    // R1 family
    int p1 = 0;
    for (int i = 0; i <= 11; i = i + 2)
    {
        if (frame[i] == 1)
            p1++;
    }
```

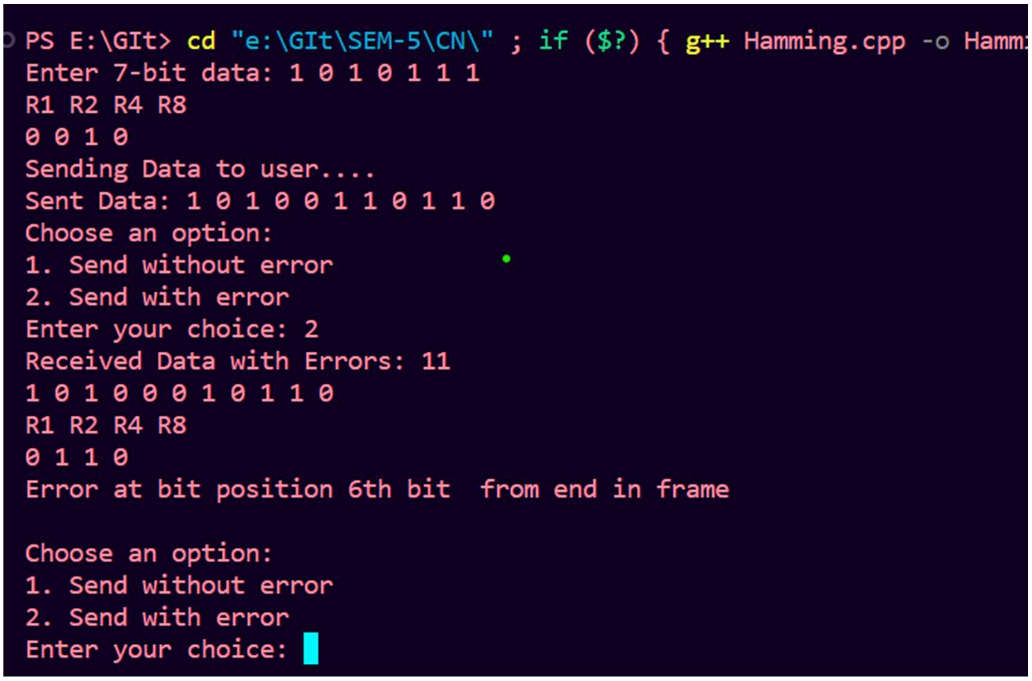


```
    R1 = 1;
    if (p1 % 2 == 0)
        R1 = 0;

    // R2 Family
    int p2 = 0;
    if (frame[9] == 1)
        p2++;
    if (frame[8] == 1)
        p2++;
    if (frame[5] == 1)
        p2++;
    if (frame[4] == 1)
        p2++;
    if (frame[1] == 1)
        p2++;
    if (frame[0] == 1)
        p2++;
    R2 = 1;
    if (p2 % 2 == 0)
        R2 = 0;

    // R4 family
    int p4 = 0;
    for (int i = 4; i <= 7; i++)
    {
        if (frame[i] == 1)
            p4++;
    }
    R4 = 1;
    if (p4 % 2 == 0)
        R4 = 0;

    // R8 family
    int p8 = 0;
    for (int i = 0; i <= 3; i++)
    {
        if (frame[i] == 1)
            p8++;
    }
    R8 = 1;
    if (p8 % 2 == 0)
        R8 = 0;
    cout << "R1 R2 R4 R8" << endl;
    cout << R8 << " " << R4 << " " << R2 << " " << R1 << endl;
}

// Function to display received data without errors
void Noerror(vector<int> frame)
{
    cout << "Received Data: ";
    for (int i = 0; i < frame.size(); i++)
    {

        cout << frame[i] << " ";
    }
    cout << "\nData is error-free (OK)" << endl;

    int R1, R2, R4, R8;

    fparity(R1, R2, R4, R8, frame);
    frame[10] = R1; //R1
    frame[9] = R2;  //R2
    frame[7] = R4;  //R4
    frame[3] = R8;  //R8
    //parity bits

    cout << "As all the parities are 0" << endl;
}

// Function to simulate received data with errors
void Witherror(vector<int> frame)
{
    vector<int> itf = {0, 1, 2, 4, 5, 6, 8};

    int randn = rand() % 7;
    if (frame[itf[randn]] == 1)
        frame[itf[randn]] = 0;
    else
        frame[itf[randn]] = 1;

    cout << "Received Data with Errors: ";
    cout<< frame.size()<<endl;
    for (int i = 0; i < frame.size(); i++)
    {
        cout << frame[i] << " ";
    }
    cout<<endl;

    int R1, R2, R4, R8;
    fparity(R1, R2, R4, R8, frame);
    frame[10] = R1; //R1
    frame[9] = R2;  //R2
    frame[7] = R4;  //R4
    frame[3] = R8;  //R8




    // Finding and displaying the error position
    int error = 0;
    if (R8 == 1)
        error += 8;
    if (R4 == 1)
        error += 4;
    if (R2 == 1)
        error += 2;
    if (R1 == 1)
        error += 1;

    cout << "Error at bit position " << error <<"th bit  from end in frame "<<
endl;
}

// Function to send the data
void sender(vector<int> &data, int &r)
{
    cout << "Enter 7-bit data: ";
    for (int i = 0; i < 7; i++)
        cin >> data[i];
    findr(r);
    int fsize = 7 + r; // m + r
    vector<int> frame(fsize, 0);

    // Initialize parity bits to -1
    frame[10] = -1; //R1
    frame[9] = -1;  //R2
    frame[7] = -1;  //R4
```



```
    frame[3] = -1;  //R8

    // Fill the frame with data
    int id = 0; // Trace the data
    for (int i = 0; i < fsize; i++)
    {
        if (frame[i] == -1)
            continue;
        frame[i] = data[id];
        id++;
    }

    // Calculate and set the parity bits
    int R1, R2, R4, R8;
    fparity(R1, R2, R4, R8, frame);
    frame[10] = R1; //R1
    frame[9] = R2;  //R2
    frame[7] = R4;  //R4
    frame[3] = R8;  //R8

    cout << "Sending Data to user...." << endl;
    cout << "Sent Data: ";
    for (int i = 0; i < fsize; i++)
    {
        cout << frame[i] << " ";
    }




while(1)
{
    cout << "\nChoose an option:\n";
cout << "1. Send without error\n";
cout << "2. Send with error\n";
cout << "Enter your choice: ";
    int choice;
  cin >> choice;
    switch (choice) {
    case 1:
        Noerror(frame);
        break;
    case 2:
        Witherror(frame);
        break;
    default:
        cout << "Invalid choice." << endl;
        break;
}
}


}

int main()
{
    srand(time(0)); // Seed the random number generator
    vector<int> data(7);
    int r; // Number of redundant bits

    sender(data, r);

    return 0;
}
```

Output:



```
PS E:\GIt> cd "e:\GIt\SEM-5\CN\" ; if ($?) { g++ Hamming.cpp -o Hamm
Enter 7-bit data: 1 0 1 0 1 1 1
R1 R2 R4 R8
0 0 1 0
Sending Data to user....
Sent Data: 1 0 1 0 0 1 1 0 1 1 0
Choose an option:
1. Send without error
2. Send with error
Enter your choice: 2
Received Data with Errors: 11
1 0 1 0 0 0 1 0 1 1 0
R1 R2 R4 R8
0 1 1 0
Error at bit position 6th bit  from end in frame

Choose an option:
1. Send without error
2. Send with error
Enter your choice: 
```