OSCN LAB – 04

Name: MOHAMMED FARNAS ALI MUDABBIR

ROLL NUMBER: 2503B05136

Write a C++ program to implement Hamming Code for error detection and correction.

Requirements:

1. Accept a binary data string as input.
2. Determine the number of **redundant bits** required.
3. Insert the redundant bits into appropriate positions (1, 2, 4, 8, …).
4. Calculate and set the **parity bits** using **even parity** logic.
5. Display the **final data packet** (data bits + redundant bits) to be transmitted.
6. Simulate the **receiver side** to:
    - Recalculate parity bits.
    - Detect if there is any **single-bit error**.
    - Display the **position of the erroneous bit** or confirm **no error** if data is correct.

Example Output:

Input data: 1011001

The data packet to be sent is: 1 0 1 0 1 1 0 1

Correct data packet received

CODE:

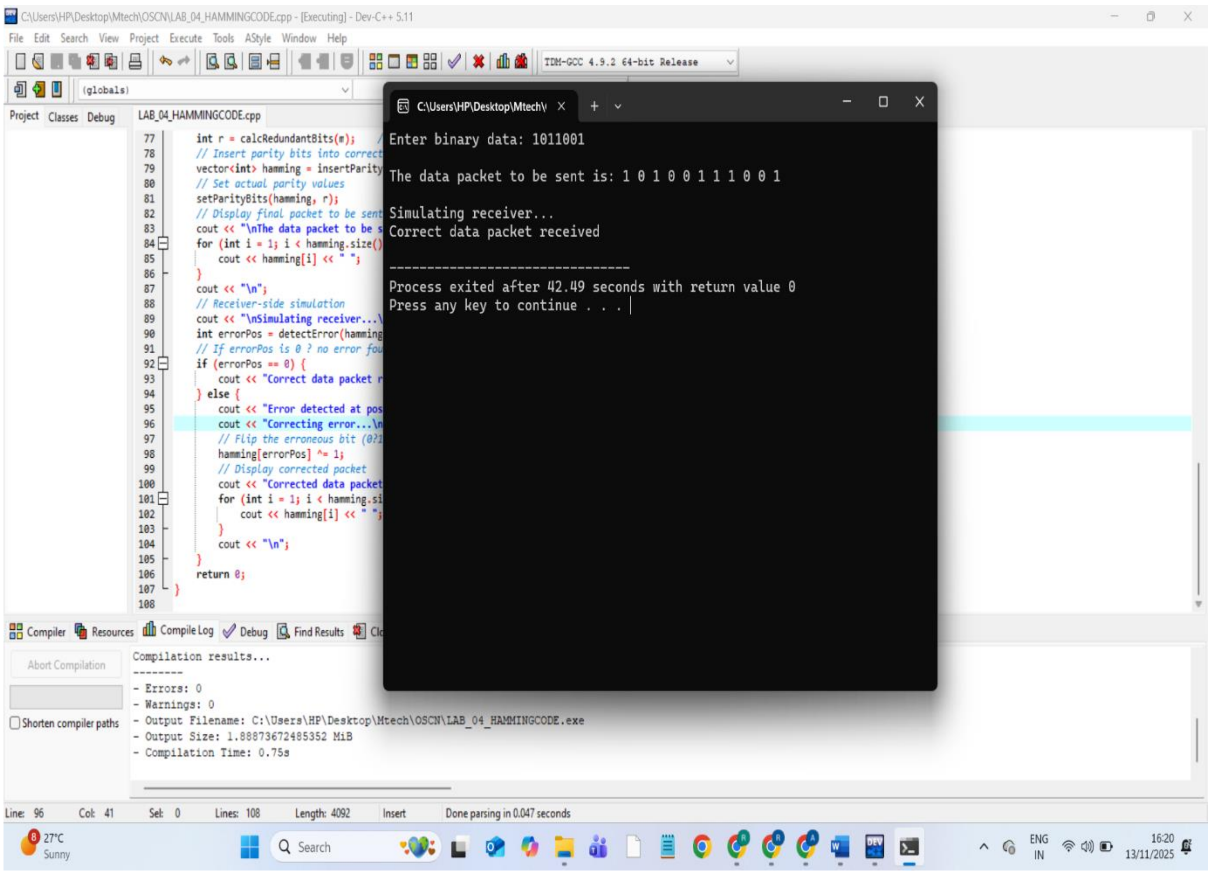
```
#include <iostream>

#include <cmath>

#include <vector>

using namespace std;
```

```
int calcRedundantBits(int m) {
    int r = 0;
    while (pow(2, r) < (m + r + 1)) {
        r++;
    }
    return r;                       // Return final redundant bits count
}

vector<int> insertParityBits(const vector<int>& data, int r) {
    int m = data.size();
    int total = m + r;
    vector<int> hamming(total + 1);
    int j = 0;                      // Pointer for data bits

    for (int i = 1; i <= total; i++) {


        if ((i & (i - 1)) == 0) {
            hamming[i] = 0;         // Set parity bit placeholder
        } else {
            hamming[i] = data[j++];  // Fill data bit
        }
    }
    return hamming;                 // Return packet with parity spots
}
// Function to calculate parity bits (EVEN parity)
void setParityBits(vector<int>& hamming, int r) {
    int total = hamming.size() - 1;  // Total bits (1-indexed)
    // For each parity bit
    for (int i = 0; i < r; i++) {
        int pos = pow(2, i);        // Position of parity bit (1,2,4,8...)
        int count = 0;              // Count of 1s for parity check
        // Check bits that influence this parity bit
        for (int j = 1; j <= total; j++) {
            if (j & pos) {          // If this bit participates in parity
                count += hamming[j]; // Add the bit value
            }
        }
        hamming[pos] = count % 2;    // Set even parity ? parity = count mod 2
    }
}
// Function to detect error at receiver side
int detectError(const vector<int>& hamming, int r) {
    int total = hamming.size() - 1;  // Total bits
    int errorPos = 0;                // Store error position
    // Recalculate parity bits
    for (int i = 0; i < r; i++) {
        int pos = pow(2, i);         // Parity position
        int count = 0;               // Count of 1s
        // Check all bits related to this parity bit
        for (int j = 1; j <= total; j++) {
            if (j & pos) {          // If bit influences this parity bit
                count += hamming[j];
            }
        }
        // If parity doesn't match (count should be even)
        if (count % 2 != 0) {
            errorPos += pos;         // Add position to error index
        }
    }
    return errorPos;                 // Return final error location
}
int main() {
```

```
string input;
cout << "Enter binary data: ";   // Ask user for binary string
cin >> input;                    // Read the input
vector<int> data;                // Vector to store bits
// Convert string characters to integers (0 or 1)
for (char c : input) {
    data.push_back(c - '0');
}
int m = data.size();             // Number of data bits
int r = calcRedundantBits(m);    // Calculate redundant bits
// Insert parity bits into correct positions
vector<int> hamming = insertParityBits(data, r);
// Set actual parity values
setParityBits(hamming, r);
// Display final packet to be sent
cout << "\nThe data packet to be sent is: ";
for (int i = 1; i < hamming.size(); i++) {
    cout << hamming[i] << " ";
}
cout << "\n";
// Receiver-side simulation
cout << "\nSimulating receiver...\n";
int errorPos = detectError(hamming, r); // Recheck parity bits
// If errorPos is 0 ? no error found
if (errorPos == 0) {
    cout << "Correct data packet received\n";
} else {
    cout << "Error detected at position: " << errorPos << "\n";
    cout << "Correcting error...\n";
    // Flip the erroneous bit (0?1 or 1?0)
    hamming[errorPos] ^= 1;
```

```
        // Display corrected packet
        cout << "Corrected data packet: ";
        for (int i = 1; i < hamming.size(); i++) {
            cout << hamming[i] << " ";
        }
        cout << "\n";
    }
    return 0;                    // End of program
}
```

OUTPUT:


C:\Users\HP\Desktop\Mtech\OSCN\LAB_04_HAMMINGCODE.cpp - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
LAB_04_HAMMINGCODE.cpp
C:\Users\HP\Desktop\Mtech\
Compiler Resources Compile Log Debug Find Results
Abort Compilation
Shorten compiler paths

```
    int r = calcRedundantBits(m);    /
    // Insert parity bits into correct
    vector<int> hamming = insertParity
    // Set actual parity values
    setParityBits(hamming, r);
    // Display final packet to be sent
    cout << "\nThe data packet to be s
    for (int i = 1; i < hamming.size()
        cout << hamming[i] << " ";
    }
    cout << "\n";
    // Receiver-side simulation
    cout << "\nSimulating receiver...\
    int errorPos = detectError(hamming
    // If errorPos is 0 ? no error fou
    if (errorPos == 0) {
        cout << "Correct data packet r
    } else {
        cout << "Error detected at pos
        cout << "Correcting error...\n
        // Flip the erroneous bit (0?1
        hamming[errorPos] ^= 1;
        // Display corrected packet
        cout << "Corrected data packet
        for (int i = 1; i < hamming.si
            cout << hamming[i] << " ";
        }
        cout << "\n";
    }
    return 0;
}
```

```
Enter binary data: 1011001

The data packet to be sent is: 1 0 1 0 0 1 1 1 0 0 1

Simulating receiver...
Correct data packet received

--------------------------------
Process exited after 42.49 seconds with return value 0
Press any key to continue . . .
```

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\HP\Desktop\Mtech\OSCN\LAB_04_HAMMINGCODE.exe
- Output Size: 1.88873672485352 MiB
- Compilation Time: 0.75s
```

Line: 96   Col: 41   Sel: 0   Lines: 108   Length: 4092   Insert   Done parsing in 0.047 seconds