Green University of Bangladesh
Department of Computer Science and Engineering(CSE)

Faculty of Sciences and Engineering

Semester: (Spring, Year:2024), B.Sc. in CSE (Day)

LAB ASSIGNMENT NO #05

Course Title: Data Communication Lab

Course Code: CSE 308   Section: 221_D3

Lab Experiment Name: Implementing of Error Detection & Correction Mechanism using Hamming Code

Student Details

Name		ID
1.	Jahidul Islam	221002504

Lab Date : 23 – 03 – 2024

Submission Date : 27 – 03 – 2024

Course Teacher's Name : Sakhaouth Hossan

[For Teachers use only: Don't Write Anything inside this box]

Lab Report Status

Marks: …………………………………   **Signature:**.....................

Comments:...   **Date:**..............................

1. TITLE OF THE LAB EXPERIMENT:

Implementing of Error Detection & Correction Mechanism using Hamming Code

2. OBJECTIVES:

After complementing this lab experiment, we will gain practical knowledge and tthe outcomes of this experiment are

- To implement the Error Detection & Correction Mechanism using Hamming Code

3. PROCEDURE:

Hamming Encoder Procedure:

Inputs:

- Binary bit stream to be encoded.

Outputs:

- Encoded data with parity bits.

Procedure Steps:

1. **Prompt User**: Display a message to prompt the user to enter a binary bit stream.
2. **Input**: Accept the binary bit stream input from the user.
3. **Calculate Parity Bits**: Determine the number of parity bits required based on the length of the input data.
4. **Encode Data**:
    - Initialize an array to store the encoded data.
    - Initialize variables for tracking positions of data bits and parity bits.
    - Iterate over each position in the encoded data array:
        - If the position is a power of 2 (parity bit position), skip and increment the parity bit tracker.
        - Otherwise, copy the corresponding data bit from the input stream to the encoded data array and increment the data bit tracker.
5. **Calculate Parity Bits**:
    - For each parity bit position:
        - Calculate the parity by counting the number of '1's in specific combinations of data bits.
        - Set the parity bit to '1' if the count of '1's is odd, otherwise set it to '0'.
6. **Display Encoded Data**: Show the encoded data with parity bits to the user

4. IMPLEMENTATION

```
#include <iostream>
#include <string>
#include <cmath>

using namespace std;

int calculateParityBits(int m) {
    int r = 0;
    while (pow(2, r) < m + r + 1) {
        r++;
    }
    return r;
}

string hammingEncode(const string& input) {
    int m = input.length();
    int r = calculateParityBits(m);

    string encodedData(m + r, '0');

    int p = 0;
    int j = 0;

    for (int i = 1; i <= m + r; i++) {
        if (i == pow(2, p)) {
            p++;
        } else {
            encodedData[i - 1] = input[j++];
        }
    }

    for (int i = 0; i < r; i++) {
        int parityIndex = pow(2, i) - 1;
        int count = 0;
        for (int j = parityIndex + 1; j <= m + r; j++) {
            if ((j & (1 << i)) != 0) {
                if (encodedData[j - 1] == '1') {
```

```
                    count++;
                }
            }
        }
        if (count % 2 != 0) {
            encodedData[parityIndex] = '1';
        }
    }

    return encodedData;
}

void displayBinaryString(const string& binaryString) {
    for (char bit : binaryString) {
        cout << bit << "";
    }
    cout << endl;
}

int main() {
    string input;
    cout << "Enter binary bit stream: ";
    cin >> input;

    string encodedData = hammingEncode(input);

    cout << "Encoded data with parity bits: ";
    displayBinaryString(encodedData);

    return 0;
}
```

5. OUTPUT

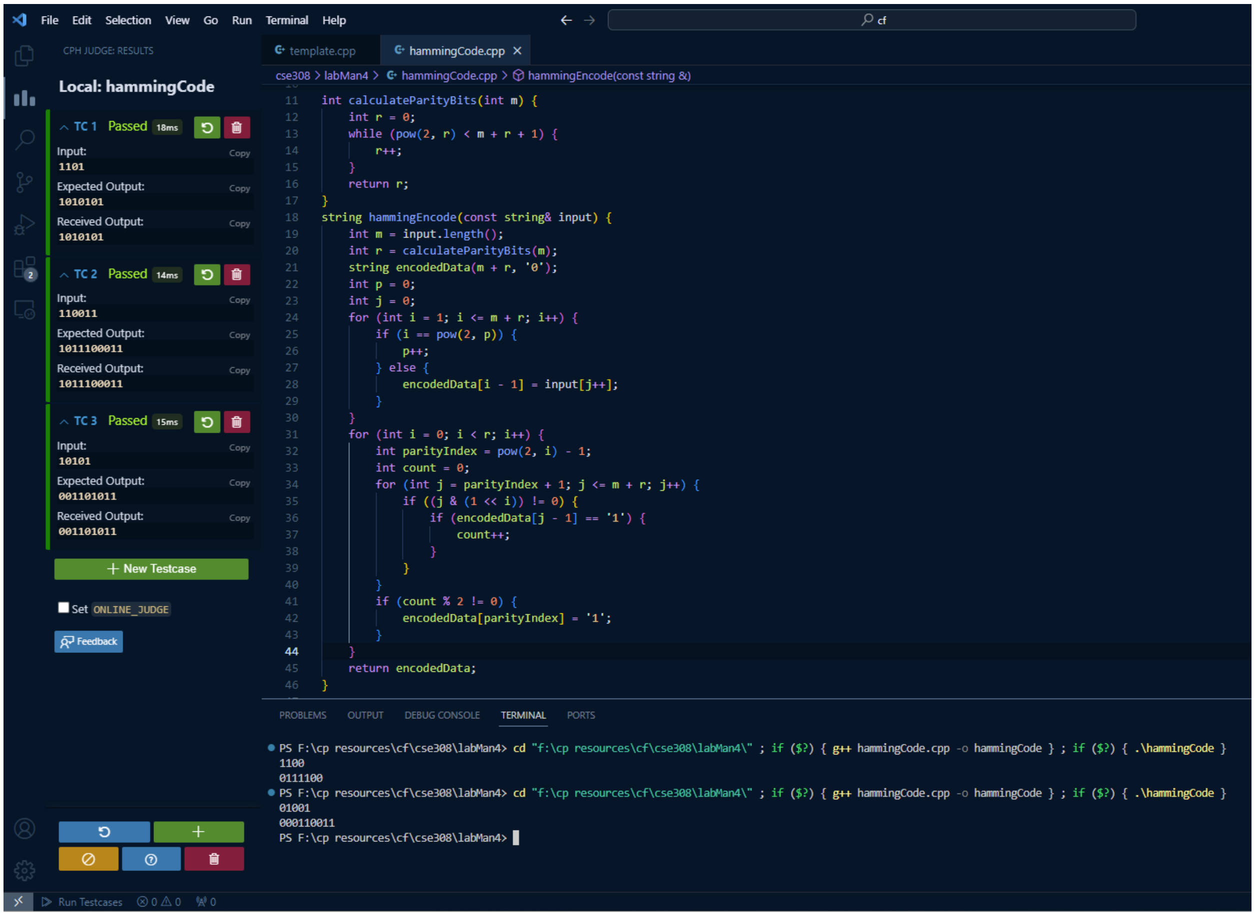
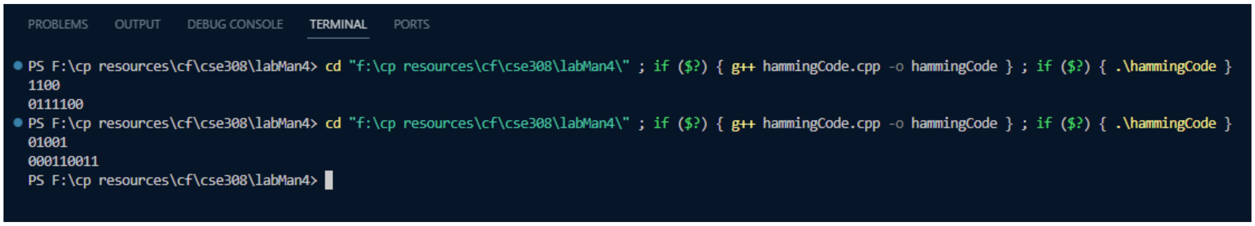


Figure 01: Shows the code and output of this code.



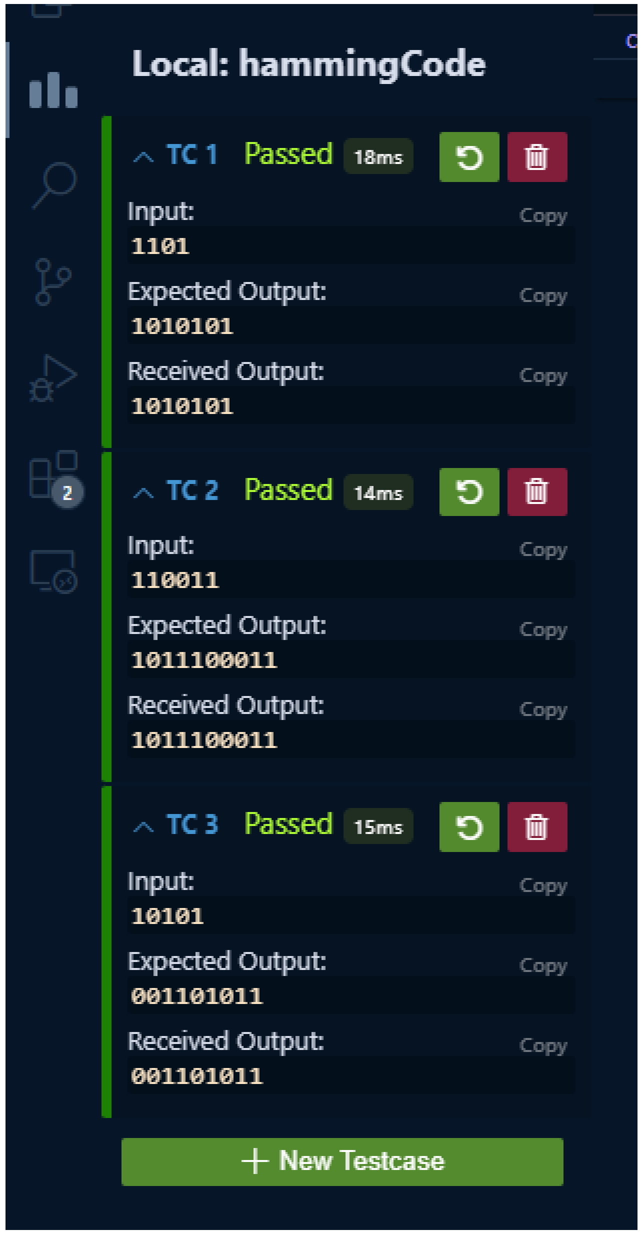


Figure 02: Output of the program.

6. ANALYSIS AND DISCUSSION:

Hamming code is effective for single-bit error detection and correction but has limitations with multiple-bit errors.