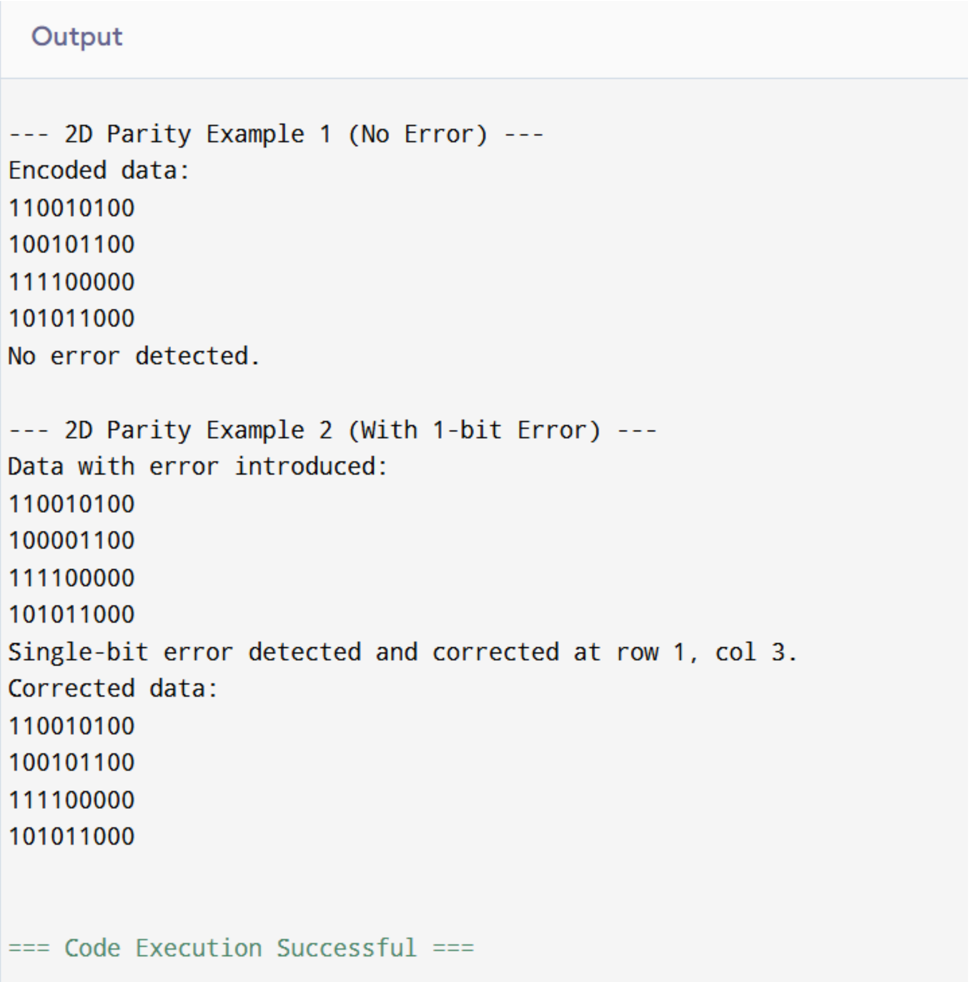
ITC Homework Assignment
Error Detection and Correction Codes

NAME - Anmol Agrawal
Roll_No - 122CS0300

1. 2D Parity Check (VRC + LRC)

```
#include <iostream>
#include <vector>
#include <algorithm> // for std::count
using namespace std;

// Function to add row (VRC) and column (LRC) parity
vector<string> add2DParity(vector<string> data) {
    // Add row parity (VRC)
    for (auto& byte : data) {
        int oneCount = count(byte.begin(), byte.end(), '1');
        byte += (oneCount % 2 == 0 ? '0' : '1');
    }

    // Add column parity (LRC)
    string lrc(data[0].size(), '0');
    for (int j = 0; j < data[0].size(); ++j) {
        int colCount = 0;
        for (auto& row : data) {
            if (row[j] == '1') colCount++;
        }
        lrc[j] = (colCount % 2 == 0 ? '0' : '1');
    }
    data.push_back(lrc);
    return data;
}

// Function to detect and correct single-bit error using 2D parity
bool detectAndCorrect2DParity(vector<string>& data) {
    int rows = data.size() - 1;
    int cols = data[0].size();

    int errorRow = -1, errorCol = -1;

    // Check row parity
    for (int i = 0; i < rows; ++i) {
        int ones = count(data[i].begin(), data[i].end(), '1');
        if (ones % 2 != 0) errorRow = i;
    }

    // Check column parity
    for (int j = 0; j < cols; ++j) {
        int ones = 0;
        for (int i = 0; i < rows; ++i) {
            if (data[i][j] == '1') ones++;
        }
        if ((ones % 2) != (data[rows][j] - '0')) errorCol = j;
    }

    if (errorRow != -1 && errorCol != -1) {
        // Single-bit error detected
        data[errorRow][errorCol] = (data[errorRow][errorCol] == '1' ? '0' : '1');
        cout << "Single-bit error detected and corrected at row " << errorRow << ", col " << errorCol << ".\n";
        return true;
    } else if (errorRow == -1 && errorCol == -1) {
        cout << "No error detected.\n";
        return true;
    } else {
        cout << "Multiple-bit error detected: cannot correct.\n";
        return false;
    }
}

void run2DParityExamples() {
    // Example 1: No error
    cout << "\n--- 2D Parity Example 1 (No Error) ---\n";
    vector<string> data1 = {"11001010", "10010110", "11110000"};
    auto encoded1 = add2DParity(data1);
    cout << "Encoded data:\n";
    for (auto& row : encoded1) cout << row << endl;
    detectAndCorrect2DParity(encoded1);

    // Example 2: With error
    cout << "\n--- 2D Parity Example 2 (With 1-bit Error) ---\n";
    vector<string> data2 = {"11001010", "10010110", "11110000"};
    auto encoded2 = add2DParity(data2);
    encoded2[1][3] = (encoded2[1][3] == '1' ? '0' : '1'); // Introduce 1-bit error
    cout << "Data with error introduced:\n";
    for (auto& row : encoded2) cout << row << endl;
    detectAndCorrect2DParity(encoded2);
    cout << "Corrected data:\n";
    for (auto& row : encoded2) cout << row << endl;
}

int main() {
    run2DParityExamples();
    return 0;
}
```

OUTPUT:



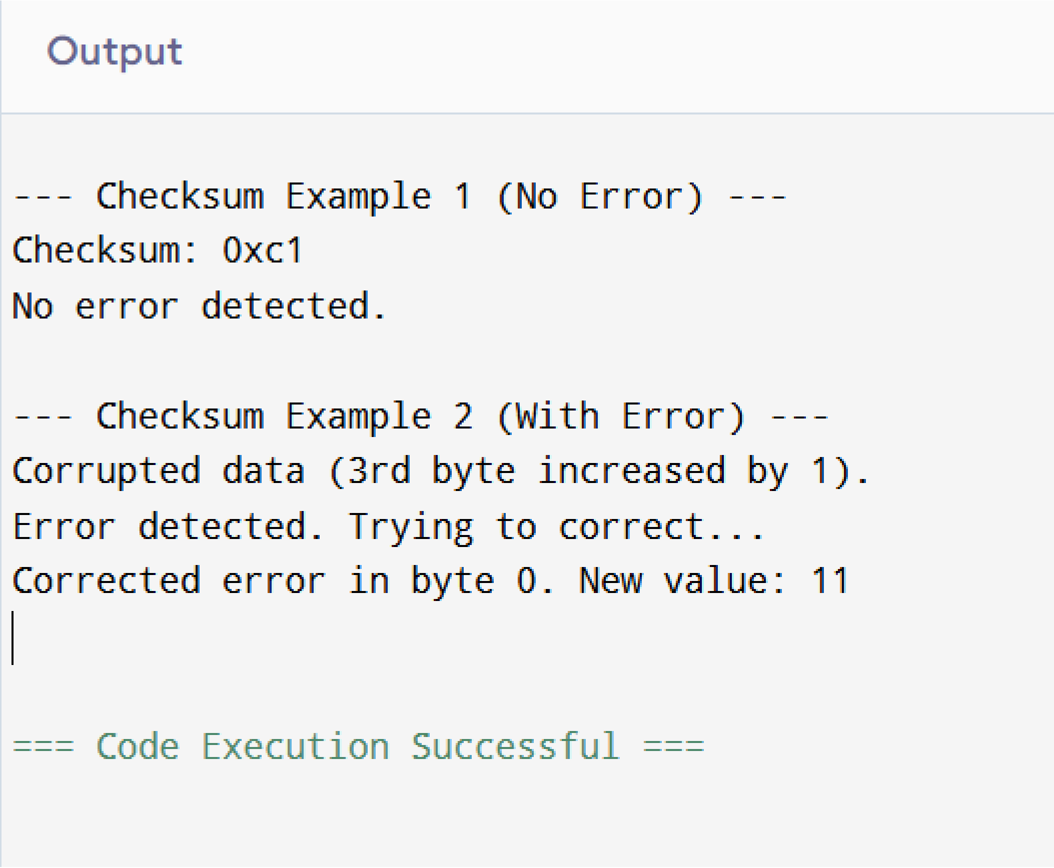
```
--- 2D Parity Example 1 (No Error) ---
Encoded data:
110010100
100101100
111100000
101011000
No error detected.

--- 2D Parity Example 2 (With 1-bit Error) ---
Data with error introduced:
110010100
100001100
111100000
101011000
Single-bit error detected and corrected at row 1, col 3.
Corrected data:
110010100
100101100
111100000
101011000


=== Code Execution Successful ===
```

2. Checksum (8-bit Two's Complement)

```
#include <iostream>
#include <vector>
using namespace std;

// Calculate checksum using 2's complement
```

```
unsigned char calculateChecksum(const vector<unsigned char>&
data) {
    unsigned int sum = 0;
    for (auto byte : data) sum += byte;
    return ~sum + 1;
}

// Try to detect and correct single-byte error
bool detectAndCorrectChecksum(vector<unsigned char>& data,
unsigned char checksum) {
    unsigned int sum = 0;
    for (auto byte : data) sum += byte;
    sum += checksum;

    if ((sum & 0xFF) == 0) {
        cout << "No error detected.\n";
        return true;
    }

    cout << "Error detected. Trying to correct...\n";

    for (int i = 0; i < data.size(); ++i) {
        unsigned char original = data[i];
        for (int delta = -10; delta <= 10; ++delta) { // Try nearby
values
            int modified = (int)original + delta;
            if (modified < 0 || modified > 255) continue;

            data[i] = (unsigned char)modified;
            unsigned int newSum = 0;
            for (auto byte : data) newSum += byte;
            newSum += checksum;

            if ((newSum & 0xFF) == 0) {
                cout << "Corrected error in byte " << i << ". New value: "
<< (int)data[i] << "\n";
                return true;
            }
        }
        data[i] = original; // Restore
    }

    cout << "Could not correct the error.\n";
    return false;
}

void runChecksumExamples() {
    // Example 1: No error
    cout << "\n--- Checksum Example 1 (No Error) ---\n";
    vector<unsigned char> data1 = {0x12, 0xA4, 0x56, 0x33};
    unsigned char checksum1 = calculateChecksum(data1);
    cout << "Checksum: 0x" << hex << (int)checksum1 << endl;
    detectAndCorrectChecksum(data1, checksum1);

    // Example 2: With error
    cout << "\n--- Checksum Example 2 (With Error) ---\n";
    vector<unsigned char> data2 = {0x12, 0xA4, 0x56, 0x33};
    unsigned char checksum2 = calculateChecksum(data2);
    data2[2] += 1; // Introduce error
    cout << "Corrupted data (3rd byte increased by 1).\n";
    detectAndCorrectChecksum(data2, checksum2);
}

int main() {
    runChecksumExamples();
    return 0;
}

```

<u>OUTPUT</u>:



```
--- Checksum Example 1 (No Error) ---
Checksum: 0xc1
No error detected.

--- Checksum Example 2 (With Error) ---
Corrupted data (3rd byte increased by 1).
Error detected. Trying to correct...
Corrected error in byte 0. New value: 11


=== Code Execution Successful ===
```