NAME : Elijah M Flomo
ROLL NO : 2506B09602

ASSIGNMENT - 05

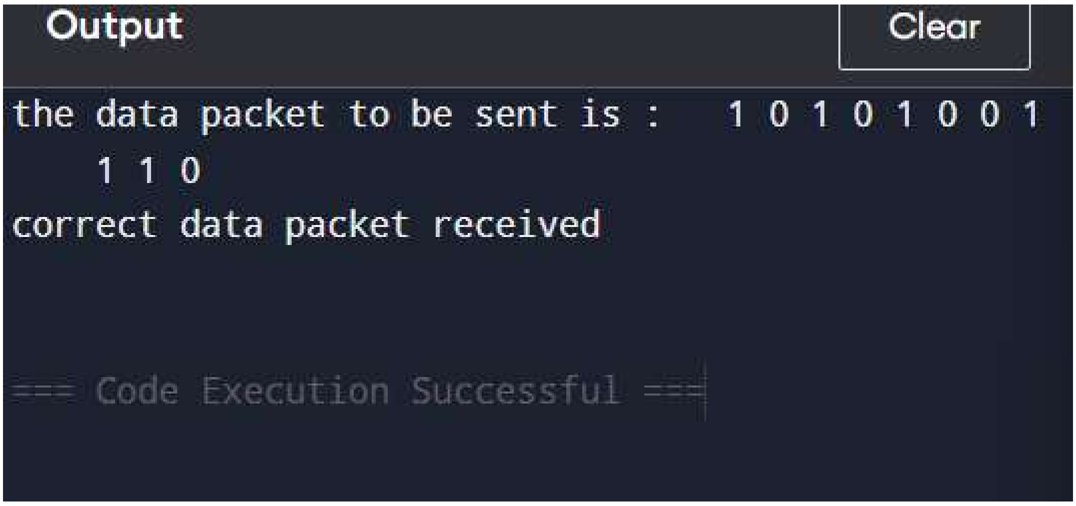
Question:

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

CODE:

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

class HammingCode {
private:
    vector<int> data;         // Original data bits
    vector<int> transmitted;  // Data + redundant bits
    int m;                    // Number of data bits
    int r;                    // Number of redundant bits

    // Calculate number of redundant bits required
    int calculateRedundantBits(int m) {
        int r = 0;
        while (pow(2, r) < (m + r + 1)) {
            r++;
        }
        return r;
    }

    // Insert redundant bits at positions that are powers of 2
    vector<int> insertRedundantBits(const vector<int>& data, int r) {
        vector<int> result(data.size() + r, 0);
        int j = 0;
        for (int i = 0; i < result.size(); i++) {
            // (i+1) = position (1-based index)
            if (((i + 1) & (i)) != 0) {
                result[i] = data[j++];
            }
        }
        return result;
    }

    // Calculate parity bits (even parity)
    void calculateParityBits(vector<int>& bits) {
        int n = bits.size();
        for (int i = 0; i < n; i++) {
            if (((i + 1) & (i)) == 0) { // Power of 2 positions
                int parity = 0;
                for (int j = 0; j < n; j++) {
                    if (((j + 1) & (i + 1)) != 0) {
                        parity ^= bits[j];
                    }
                }
                bits[i] = parity; // even parity
            }
        }
    }

    // Calculate syndrome to check for errors at receiver side
    int calculateSyndrome(const vector<int>& received) {
        int syndrome = 0;
        int n = received.size();
        for (int i = 0; i < n; i++) {
            if (((i + 1) & (i)) == 0) {
                int parity = 0;
                for (int j = 0; j < n; j++) {
                    if (((j + 1) & (i + 1)) != 0) {
                        parity ^= received[j];
                    }
                }
                if (parity != 0) {
                    syndrome += (i + 1);
                }
            }
        }
        return syndrome;
    }

public:
    // Constructor
    HammingCode(string input) {
        for (int i = 0; i < input.length(); i++) {
            data.push_back(input[i] - '0');
        }
        m = data.size();
        r = calculateRedundantBits(m);
        transmitted = insertRedundantBits(data, r);
        calculateParityBits(transmitted);
    }

    // Display transmitted packet
    void displayTransmittedData() {
        cout << "\nThe data packet to be sent is: ";
        for (int i = 0; i < transmitted.size(); i++) {
            cout << transmitted[i] << " ";
        }
        cout << endl;
    }
    // Simulate receiver side (check error)
    void simulateReceiver(const vector<int>& received) {
        int syndrome = calculateSyndrome(received);
        if (syndrome == 0) {
            cout << "\nCorrect data packet received.\n";
        } else {
            cout << "\nError detected at position: " << syndrome << endl;
            vector<int> corrected = received;
            corrected[syndrome - 1] ^= 1;
            cout << "Corrected data packet: ";
            for (int i = 0; i < corrected.size(); i++) {
                cout << corrected[i] << " ";
            }
            cout << endl;
        }
    }

    vector<int> getTransmittedData() {
        return transmitted;
    }
};

int main() {
    string input;
    cout << "Input data: ";
    cin >> input;

    HammingCode hc(input);
    hc.displayTransmittedData();

    vector<int> transmitted = hc.getTransmittedData();
    hc.simulateReceiver(transmitted);
    return 0;
}
```

OUTPUT:



```
the data packet to be sent is :   1 0 1 0 1 0 0 1
    1 1 0
correct data packet received


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