```
// AP21110010201
// Perumalla Dharan
// Write a program to implement hamming codes for error detection and correction.


#include <iostream>
using namespace std;
int calculateRedundancyBits(int dataBits) {
    int r = 0;
    while ((1 << r) < dataBits + r + 1) {
        r++;
    }
    return r;
}
void calculateParityBits(int code[], int size) {
    int r = calculateRedundancyBits(size - 1);

    for (int i = 0; i < r; i++) {
        int parityBitPosition = (1 << i) - 1;
        code[parityBitPosition] = 0;

        for (int j = parityBitPosition + 1; j < size; j++) {
            if (((j + 1) >> i) & 1) {
                code[parityBitPosition] ^= code[j];
            }
        }
    }
}
int main() {
    int numDataBits;
    cout << "Enter the number of data bits: ";
    cin>> numDataBits;
    int numTotalBits = numDataBits + calculateRedundancyBits(numDataBits);
    int receivedCode[numTotalBits] = {0};
    cout << "Enter received code (" << numTotalBits << " bits): ";
    for (int i = 0; i < numTotalBits; i++) {
        cin >> receivedCode[i];
    }
    int r = calculateRedundancyBits(numDataBits);
    int errorPositions[32] = {0};
    int errorCount = 0;

    for (int position = 1; position <= numTotalBits; position *= 2) {
        int parityBitPosition = position - 1;
```

```
        int calculatedParity = 0;

        for (int j = parityBitPosition; j < numTotalBits; j++) {
            if (((j + 1) >> (position - 1)) & 1) {
                calculatedParity ^= receivedCode[j];
            }
        }

        if (receivedCode[parityBitPosition] != calculatedParity) {
            errorPositions[errorCount++] = position;
        }
    }

    if (errorCount == 0) {
        cout << "No errors found in the received message." << endl;
    } else {
        cout << "Errors detected at bit positions: ";
        for (int i = 0; i < errorCount; i++) {
            cout << errorPositions[i] << " ";
        }
        cout << endl;

        cout << "Corrected code: ";
        for (int i = 0; i < numTotalBits; i++) {
            int isPositionError = false;
            for (int j = 0; j < errorCount; j++) {
                if (errorPositions[j] == (i + 1)) {
                    isPositionError = true;
                    break;
                }
            }
            if (isPositionError) {
                receivedCode[i+1] = 1 - receivedCode[i+1];
            }
            cout << receivedCode[i] << " ";
        }
        cout << endl;
    }

    return 0;
}
```