```cpp
#include <iostream>
#include <bitset>

using namespace std;

// Function to calculate Hamming Code
string hammingCode(string data) {
    // Calculate number of parity bits required
    int r = 0;
    while ((1 << r) < data.length() + r + 1) {
        r++;
    }

    // Create a vector to store encoded bits
    string encoded = "";

    // Initialize parity bits to 0
    for (int i = 0; i < r; i++) {
        encoded += '0';
    }

    int j = 0; // Index for data bits
    int k = 0; // Index for parity bits

    // Traverse the encoded string and set parity bits
    for (int i = 1; i <= encoded.length(); i++) {
        if ((i & (i - 1)) == 0) { // If i is a power of 2 (parity bit)
            encoded[i - 1] = '0'; // Initialize parity bit to 0
        } else {
            encoded[i - 1] = data[j++]; // Set data bit
        }
    }

    // Calculate parity bits
    for (int i = 0; i < r; i++) {
        int parity = 0;
        for (int j = (1 << i); j <= encoded.length(); j += (1 << (i + 1)))
{
            for (int k = 0; k < (1 << i) && (j + k) <= encoded.length();
k++) {
```

```
                parity ^= (encoded[j + k - 1] - '0'); // Calculate XOR of
bits
            }
        }
        encoded[(1 << i) - 1] = parity + '0'; // Set parity bit
    }

    return encoded;
}

// Function to detect errors in Hamming Code
int detectError(string encoded) {
    int r = 0;
    while ((1 << r) < encoded.length()) {
        r++;
    }

    int errorBit = 0;
    for (int i = 0; i < r; i++) {
        int parity = 0;
        for (int j = (1 << i); j <= encoded.length(); j += (1 << (i + 1)))
{
            for (int k = 0; k < (1 << i) && (j + k) <= encoded.length();
k++) {
                parity ^= (encoded[j + k - 1] - '0'); // Calculate XOR of
bits
            }
        }
        errorBit += parity << i;
    }

    return errorBit;
}

int main() {
    string data;
    cout << "Enter 8-bit data: ";
    cin >> data;

    if (data.length() != 8) {
```

```
        cout << "Invalid input! Please enter exactly 8 bits." << endl;
        return 1;
    }

    string encoded = hammingCode(data);
    cout << "Hamming Code: " << encoded << endl;

    cout << "Enter received code: ";
    string received;
    cin >> received;

    int errorBit = detectError(received);
    if (errorBit != 0) {
        cout << "Error detected at bit position: " << errorBit << endl;
    } else {
        cout << "No errors detected." << endl;
    }

    return 0;
}
```