Assignment-A4 (Error Detection & Correction) – Output

Code

CRC

```cpp
#include <iostream>
#include <cstring>
using namespace std;
string XOR(string data, string key) {
    // Dividend is data
    // Divisor is the primary key, i.e. the key
    string result = "";
    int dataLen = data.length();
    int keyLen = key.length();
    // Perform XOR operation
    for (int i=0; i<keyLen; i++) {
        if (i < dataLen) {
            // Only perform XOR if within the length of data
            if (data[i] == key[i]) {
                result += '0';
            }
            else {
                result += '1';
            }
        } else {
            // If data length exceeded, append the key
            result += key[i];
        }
    }
    return result;
}
string encoder(string data, string key) {
    int keyLen = key.length();
    // Append n-1 zeroes to the data
    string dataWithZeroes = data + string(keyLen-1, '0');
    // Perform XOR operation with the key
    string remainder = XOR(dataWithZeroes, key);
    // Get the remainder (last n-1 bits)
    string crc = remainder.substr(remainder.length() - (keyLen-1));
    // Append the CRC to the original data
    return data + crc;
```

```cpp
}
string performDivision(string data, string key) {
    int keyLen = key.length();
    // Initialize with the data to be divided
    string temp = data.substr(0, keyLen);
    // Perform XOR operations on each segment
    for (int i = keyLen; i < data.length(); i++) {
        if (temp[0] == '1') {  // Only perform XOR if the leading bit is
1
            temp = XOR(temp, key);
        }
        temp = temp.substr(1) + data[i];  // Shift left and add the next
bit
    }
    // Perform the final XOR operation
    if (temp[0] == '1') {
        temp = XOR(temp, key);
    }
    // Extract the remainder (last keyLen-1 bits)
    return temp.substr(temp.length() - (keyLen - 1));
}
// Function to check the correctness of received data
bool checkData(string data, string key) {
    string remainder = performDivision(data, key);
    return (remainder.find('1') == string::npos); // No '1' means
remainder is all zeros
}
int main() {
    string data, key;
    cout << endl << "Enter data:\t";
    getline(cin, data);
    cout << "Enter primary key:\t";
    getline(cin, key);
    cout<<endl<<"Original data:\t"<<data;
    cout<<endl<<"Key:\t"<<key;
    string messageToSend = encoder(data, key);
    cout<<endl<<"------------------------------------"<<endl;
    cout<<"Message to be sent:\t"<<messageToSend;
    cout<<endl<<"------------------------------------"<<endl;
    string receivedData;
    cout<<endl<<"HINT: Received data should be the same as message to be
sent.";
    cout<<endl<<"Enter received data:\t";
    getline(cin, receivedData);
```

```cpp
    if (receivedData == messageToSend) {
        cout<<"The received data is correct."<<endl;
    } else {
        cout<<"The received data appears to be tampered."<<endl;
    }
    return 0;
}
```

Hamming Code

```cpp
#include <iostream>
#include <cmath>
#include <vector>
using namespace std;
// Function to calculate the number of parity bits needed
int calculateParityBits(int dataBits) {
    int parityBits = 0;
    while (pow(2, parityBits) < dataBits + parityBits + 1) {
        parityBits++;
    }
    return parityBits;
}
// Function to encode the data using Hamming code
vector<int> encodeData(vector<int> data) {
    int dataBits = data.size();
    int parityBits = calculateParityBits(dataBits);
    vector<int> encoded(dataBits + parityBits, 0);
    // Set the data bits
    int j = 0;
    for (int i = 0; i < encoded.size(); i++) {
        if (i + 1 == pow(2, j)) {
            j++;
        } else {
            encoded[i] = data[i - j];
        }
    }
    // Calculate and set the parity bits
    for (int i = 0; i < parityBits; i++) {
        int parityBit = pow(2, i);
        int sum = 0;
        for (int j = parityBit - 1; j < encoded.size(); j += 2 *
parityBit) {
            for (int k = 0; k < parityBit; k++) {
```

```cpp
                    if (j + k < encoded.size()) {
                        sum += encoded[j + k];
                    }
                }
            }
            encoded[parityBit - 1] = sum % 2;
        }
    }
    return encoded;
}


// Function to check for errors in the encoded data
int checkForErrors(vector<int> encoded) {
    int parityBits = calculateParityBits(encoded.size() -
calculateParityBits(encoded.size()));
    int errorPosition = 0;
    for (int i = 0; i < parityBits; i++) {
        int parityBit = pow(2, i);
        int sum = 0;
        for (int j = parityBit - 1; j < encoded.size(); j += 2 *
parityBit) {
            for (int k = 0; k < parityBit; k++) {
                if (j + k < encoded.size()) {
                    sum += encoded[j + k];
                }
            }
        }
        errorPosition += (sum % 2) * parityBit;
    }
    return errorPosition;
}
int main() {
    int dataBits;
    cout<<"Enter the number of data bits:\t";
    cin >> dataBits;
    vector<int> data(dataBits);
    cout<<endl<<"NOTE: Make sure the bits are entered in binary format,
separated by spaces.\nEg. 1 0 0 1 (for 4 data bits).";
    cout<<endl<<"Enter the data bits:\t";
    for (int i = 0; i < dataBits; i++) {
        cin >> data[i];
    }
    vector<int> encoded = encodeData(data);
    cout<<endl<<"--------------------"<<endl;
    cout<<"Encoded bits are:\t";
```

```cpp
    for (int bit : encoded) {
        cout << bit << " ";
    }
    cout<<endl<<"--------------------"<<endl;
    cout<<endl<<"Enter the encoded bits:\t";
    vector<int> receivedEncoded(encoded.size());
    for (int i = 0; i < encoded.size(); i++) {
        cin >> receivedEncoded[i];
    }
    int errorPosition = checkForErrors(receivedEncoded);
    if (errorPosition == 0) {
        cout<<"No errors detected."<<endl;
    } else {
        cout<<"Error detected at position: "<<errorPosition<<endl;
    }
    return 0;
}
```

Output

CRC


```
[overnion - Codes (/run/media/overnion/persistence/Fil
$ g++ Code-A5\ \(Sliding\ Window\).cpp && ./a.out
Enter the window size:  3
Enter the total number of frames:       5
Sent frame 0
Received frame 1, but expected frame 0
Sent frame 1
Received frame 3, but expected frame 0
Sent frame 2
Received frame 2, but expected frame 0
Window is full. Waiting before sending next frames.
Received frame 2, but expected frame 0
Window is full. Waiting before sending next frames.
Received frame 4, but expected frame 0
Received frame 1, but expected frame 0
Window is full. Waiting before sending next frames.
Window is full. Waiting before sending next frames.
Received frame 1, but expected frame 0
Received frame 3, but expected frame 0
Window is full. Waiting before sending next frames.
Window is full. Waiting before sending next frames.
Received frame 0
Received frame 3, but expected frame 1
Sent frame 3
Received frame 4, but expected frame 1
Window is full. Waiting before sending next frames.
Window is full. Waiting before sending next frames.
Window is full. Waiting before sending next frames.
Received frame 2, but expected frame 1
Window is full. Waiting before sending next frames.
Received frame 1
Sent frame 4
```

```
Received frame 4, but expected frame 2
Window is full. Waiting before sending next frames.
Received frame 4, but expected frame 2
Received frame 0, but expected frame 2
Window is full. Waiting before sending next frames.
Received frame 1, but expected frame 2
Received frame 1, but expected frame 2
Window is full. Waiting before sending next frames.
Received frame 0, but expected frame 2
Window is full. Waiting before sending next frames.
Received frame 0, but expected frame 2
Window is full. Waiting before sending next frames.
Received frame 2
Received frame 0, but expected frame 3
Window is full. Waiting before sending next frames.
Window is full. Waiting before sending next frames.
Received frame 2, but expected frame 3
Window is full. Waiting before sending next frames.
Window is full. Waiting before sending next frames.
Window is full. Waiting before sending next frames.
Window is full. Waiting before sending next frames.
Received frame 2, but expected frame 3
Received frame 4, but expected frame 3
Window is full. Waiting before sending next frames.
Received frame 2, but expected frame 3
Window is full. Waiting before sending next frames.
Window is full. Waiting before sending next frames.
Received frame 3
Window is full. Waiting before sending next frames.
Window is full. Waiting before sending next frames.
Received frame 4
All frames sent and received successfully!
```

Hamming Code

```
[overnion - Codes (/run/media/overnion/persistence/Files/git/sppu-te-comp-cc
$ g++ Code-A4\ \(Hamming\ Code\).cpp && ./a.out
Enter the number of data bits:  4

NOTE: Make sure the bits are entered in binary format, separated by spaces.
Eg. 1 0 0 1 (for 4 data bits).
Enter the data bits:    1 0 1 1


--------------------
Encoded bits are:      0 1 1 0 0 1 1
--------------------


Enter the encoded bits: 0 1 1 0 0 1 1
No errors detected.
[overnion - Codes (/run/media/overnion/persistence/Files/git/sppu-te-comp-cc
$ g++ Code-A4\ \(Hamming\ Code\).cpp && ./a.out
Enter the number of data bits:  4

NOTE: Make sure the bits are entered in binary format, separated by spaces.
Eg. 1 0 0 1 (for 4 data bits).
Enter the data bits:    1 0 1 1


--------------------
Encoded bits are:      0 1 1 0 0 1 1
--------------------


Enter the encoded bits: 0 1 1 0 1 0 0
Error detected at position: 4
```