Checksum

#include <iostream>

#include <string>

#include <cmath>

using namespace std;

int To\_Decimal(string data) {

    int n = data.length(); int num = 0;

    for (int i = 0; i < n; i++) {

        num = num \* 2 + (data[i] - '0');

    }

    return num;

}

string To\_Binary(int num, int data\_len) {

    string data = "";

    while (num > 0) {

        data = to\_string(num % 2) + data; num = num / 2;

    }

    int n = data.length();

    if (n < data\_len) {

        data = string(data\_len - n, '0') + data;

    }

    return data;

}

string complement(string data) {

    int n = data.length();

    for (int i = 0; i < n; i++) {

        if (data[i] == '0') {

            data[i] = '1';

        } else {

            data[i] = '0';

        }

    }

    return data;

}

void errorcheck(string data1, string data2, string checksum)

{

    int Final = To\_Decimal(data1) + To\_Decimal(data2) + To\_Decimal(checksum);

    if (Final > pow(2, data1.length()) - 1) {

        Final = Final - (pow(2, data1.length()) - 1);

    }

    string To\_Send = To\_Binary(Final, data1.length());

    To\_Send = complement(To\_Send);

    for (int i = 0; i < To\_Send.length(); i++) {

        if (To\_Send[i] != '0') {

            cout << "\nError in data\n"; return;

        }

    }

    cout << "\nNo error in data\n";

}

int main(void) {

    string data1, data2;

    int data\_len = data1.length();

    cout << "Enter data 1 : "; cin >> data1;

    cout << "Enter data 2 : "; cin >> data2;

    int num1 = To\_Decimal(data1);

    int num2 = To\_Decimal(data2);

    int sum = num1 + num2;

    if (sum > pow(2, data\_len) - 1) {

        sum = sum - (pow(2, data\_len) - 1);

    }

    string data = To\_Binary(sum, data\_len);

    string checksum = complement(data);

    cout << "\nChecksum : " << checksum << endl;

    errorcheck(data1, data2, checksum);

    return 0;

}

Crc

#include<iostream>

using namespace std;

string XOR(string a , string b) {

    string result = ""; int n = b.length();

    for(int i =1;i<n;i++) {

        if(a[i] == b[i]) {

            result += "0";

        } else {

            result += "1";

        }

    }

    return result;

}

void division(string data , string gen ,int data\_len , int gen\_len) {

    int len = gen\_len;

    string temp = data.substr(0,len); int n = data\_len;

    while(len < n) {

        if(temp[0] == '1') {

            temp = XOR(gen , temp) + data[len];

        } else {

            temp = XOR(string(len , '0') ,temp) + data[len];

        }

        len += 1;

    }

    if(temp[0] == '1') {

        temp = XOR(gen , temp);

    } else {

        temp = XOR(string(len , '0') , temp);

    }

    cout<<"\n\nRemainder is : "<<temp; int new\_len = data\_len - gen\_len + 1;

    string new\_data= data;

    for(int i =new\_len;i<data\_len;i++) {

        new\_data[i] = temp[i - new\_len];

    }

    cout<<"\n\nData to be sent is : "<<new\_data;

    cout<<endl;

}

int main(void) {

    string data , generator;cout<<"Enter data to be sent : ";

    cin >> data;

    cout<<"Enter Generator : "; cin>>generator;

    int gen\_len = generator.length();

    for(int i = 0;i< gen\_len -1 ;i++) {

        data += '0';

    }

    int data\_len = data.length();

    division(data , generator , data\_len , gen\_len);

}

Nrz

#include <bits/stdc++.h>

using namespace std;

int main() {

    int n;

    cout << "Enter the number of bits: ";

    cin >> n;

    vector<int> arr(n);

    for (int i = 0; i < n; i++) {

        int x;

        cin >> x;

        arr[i] = x;

    }

    // Calculate the width for each bit representation

    const int bit\_width = 5; // Adjust width for symmetry

    // Printing all the highs

    for (int i = 0; i < n; i++) {

        if (arr[i] == 1)

            cout << setw(bit\_width) << "high";

        else

            cout << setw(bit\_width) << " ";

    }

    cout << endl;

    // Print the waveform

    for (int i = 0; i < n; i++) {

        if (arr[i] == 0)

            cout << string(bit\_width, '\_');

        else

            cout << string(bit\_width, '\u203E');

        // Print the transition only if it's not the last bit

        if (i != n - 1) {

            if (arr[i] != arr[i + 1])

                cout << "|";

            else

                cout << " ";

        }

    }

    cout << endl;

    // Printing all the lows

    for (int i = 0; i < n; i++) {

        if (arr[i] == 0)

            cout << setw(bit\_width) << "low";

        else

            cout << setw(bit\_width) << " ";

    }

    cout << endl;

    return 0;

}

/\*

Sample Input:

8

0 0 1 1 0 1 0 0

\*/

Manchester

#include <bits/stdc++.h>

using namespace std;

int main() {

    int n;

    cout << "Enter the number of bits: ";

    cin >> n;

    vector<int> arr(n);

    for (int i = 0; i < n; i++) {

        int x;

        cin >> x;

        arr[i] = x;

    }

    const int bit\_width = 5; // Width for each bit representation

    // Print the waveform

    for (int i = 0; i < n; i++) {

        if (arr[i] == 0)

            cout << setw(bit\_width) << "\_\_|\u203E\u203E";

        else

            cout << setw(bit\_width) << "\u203E|\_\_";

    }

    cout << endl;

    return 0;

}

/\*

Sample Input:

8

0 0 1 1 0 1 0 0

\*/

Differential Manchester

#include <bits/stdc++.h>

using namespace std;

int main()

{

    int n;

    cout << "Enter the number of bits: ";

    cin >> n;

    vector<int> arr(n);

    for (int i = 0; i < n; i++)

    {

        int x;

        cin >> x;

        arr[i] = x;

    }

    // Initial waveform printing for the first bit

    if (arr[0] == 0)

        cout << "|\_\_|";

    else

        cout << "\_\_|\u203E\u203E";

    // Print the rest of the waveform based on transitions

    for (int i = 1; i < n; i++)

    {

        if (arr[i] == 0)

        {

            if (arr[i - 1] == 1)

                cout << "|\_\_|";

            else

                cout << "  \_\_|";

        }

        else

        {

            if (arr[i - 1] == 0)

                cout << "\u203E\u203E|";

            else

                cout << "|\_\_";

        }

    }

    cout << endl;

    return 0;

}

/\*

Sample Input:

8

0 0 1 1 0 1 0 0

\*/

Bit

#include <iostream>

using namespace std;

void bit\_stuffing(string &data) {

    int i = 0, count = 0;

    while (i < data.length()) {

        if (data[i] == '1') {

            count++;

        } else {

            count = 0;

        }

        if (count == 5) {

            if (i + 1 < data.length() && data[i + 1] == '1') {

                data.insert(i + 1, 1, '0');

                i++; // Move past the inserted '0'

            }

            count = 0; // Reset the count after stuffing

        }

        i++;

    }

    cout << "Stuffed data: " << data << endl;

}

int main() {

    string data;

    cout << "Enter the data: ";

    cin >> data;

    bit\_stuffing(data);

    return 0;

}

Byte

#include <iostream>

using namespace std;

int main(void) {

    string data; string d; int n;

    cout << "Enter the number of bytes: "; cin >> n;

    data += "FLAG ";

    for (int i = 0; i < n; i++) {

        cout << "Enter the data: "; cin >> d;

        if (d == "ESC" || d == "FLAG") {

            data += "ESC ";

        }

        data += d;

        data += " ";

    }

    data += "FLAG ";

    cout << "Stuffed data:  " << data << endl; return 0;

}

Stop and wait

#include <iostream>

#include <string>

#include <cstdlib>

#include <unistd.h> // For sleep function

using namespace std;

void sender(string data);

void receiver(string data);

bool ack\_received = false; // Global variable to simulate acknowledgment

int main() {

    string data;

    cout << "Enter data to be sent: ";

    cin >> data;

    // Start sender and receiver in the main thread

    sender(data);

    return 0;

}

void sender(string data) {

    int frame = 0;

    while (frame < data.length()) {

        cout << "Sending frame: " << data[frame] << endl;

        sleep(1); // Simulate transmission delay

        // Simulate sending frame to receiver

        receiver(data.substr(frame, 1));

        // Wait for acknowledgment

        if (ack\_received) {

            cout << "Acknowledgment received for frame: " << data[frame] << endl;

            ack\_received = false; // Reset acknowledgment flag

            frame++;

        } else {

            cout << "Acknowledgment not received. Resending frame: " << data[frame] << endl;

        }

    }

    cout << "All frames sent successfully." << endl;

}

void receiver(string data) {

    // Simulate processing delay

    sleep(1);

    // Randomly determine if the acknowledgment will be sent

    if (rand() % 2 == 0) {

        cout << "Frame received: " << data << endl;

        ack\_received = true;

    } else {

        cout << "Frame lost: " << data << endl;

        ack\_received = false;

    }

}

Socket

Client

import java.io.\*;

import java.net.\*;

public class Client {

public static void main(String[] args) {

String serverAddress = "127.0.0.1"; // Server's IP address

int serverPort = 12345; // Server's port number

try {

// Create a socket connecting to the server

Socket socket = new Socket(serverAddress, serverPort);

// Get the output stream of the socket

OutputStream outputStream = socket.getOutputStream();

// Create a PrintWriter to write messages to the output stream

PrintWriter out = new PrintWriter(outputStream, true);

// Send a message to the server

out.println("Hello, server!");

// Close the socket

socket.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

Server:

import java.io.\*;

import java.net.\*;

public class Server {

public static void main(String[] args) {

int portNumber = 12345; // Port number the server will listen on

try {

// Create a ServerSocket object

ServerSocket serverSocket = new ServerSocket(portNumber);

System.out.println("Server is running and waiting for client connection...");

// Wait for a client connection

Socket clientSocket = serverSocket.accept();

System.out.println("Client connected: " + clientSocket.getInetAddress());

// Get the input stream of the client socket

InputStream inputStream = clientSocket.getInputStream();

BufferedReader in = new BufferedReader(new InputStreamReader(inputStream));

// Read the message from the client

String message = in.readLine();

System.out.println("Received message from client: " + message);

// Close the sockets

clientSocket.close();

serverSocket.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}