



Green University of Bangladesh
Department of Computer Science and Engineering (CSE)
Faculty of Sciences and Engineering
Semester: (Spring, Year:2021), B.Sc. in CSE (Day/Eve)

Course Title: Data Communication
Course Code: CSE 308
Section: DC

Lab Project Name: Error-Detection and Correction

Student Details

Name		ID
1.	Zahid Hasan	201902060

Submission Date : 08/09/2022
Course Teacher's Name : Mr. Abdullah Al Arif

[For Teachers use only: **Don't Write Anything inside this box**]

Lab Project Status

Marks:

Signature:

Comments:

Date:

Table of Contents

Chapter 1 Introduction	3
1.1 Introduction.....	3
1.2 Design Goals/Objective	3
Chapter 2 Design/Development/Implementation of the Project.....	Error! Bookmark not defined.
2.1 Section (Choose the name of this section as appropriate with your project).....	Error! Bookmark not defined.
2.2 Section (Choose the name of this section as appropriate with your project).....	Error! Bookmark not defined.
2.2.1 Subsection	Error! Bookmark not defined.
Chapter 3 Performance Evaluation	Error! Bookmark not defined.
3.1 Simulation Environment/ Simulation Procedure.....	Error! Bookmark not defined.
3.2 Results and Discussions	8
Chapter 4 Conclusion	9
4.1 Introduction.....	9
4.1 Practical Implications	Error! Bookmark not defined.
4.2 Scope of Future Work.....	9
References	9

Chapter 1

Introduction

1.1 Introduction

This Simulation can visualize the concept of a few data communication topics Implementation of bit stuffing & De-stuffing, Character stuffing and De-stuffing, Parity Check Code, Cyclic Redundancy Code, Hamming Code. This Simulation visualize based on users' given data. It will be helpful for students to understand those topics and it will be easy to learn for them.

1.2 Design Goals/Objective

When we are trying to learn something we learn by reading, and listening. But I think isn't an idle way to learn cause we read and we listen but we don't know how's works in reality. If we want to learn something properly, we must understand how those things work behind reality. This simulation will show you how's work stuffing, de-stuffing, and hamming code and helps you to understand the concept.

1.3 TOOLS & TECHNOLOGIES

We have used to make this project:

- IDE- Codeblocks
- C++ programming Language

Chapter 2

Design/Development/Implementation of the Project

2.1 User-Module

1. Bit Stuffing and De-Stuffing
2. Character Stuffing and De-Stuffing
3. Parity Check
4. Cyclic Redundancy Check
5. Hamming Code
7. Exit. Thank You

2.2 Implementation

Bit Stuffing and De-Stuffing

Bit Stuffing is a process of inserting an extra bit as 0, once the frame sequence encountered 5 consecutive 1's. Given an array, `arr[]` of size N consisting of 0's and 1's, the task is to return an array after the bit stuffing.

Bit De-stuffing is the complete opposite of Bit Stuffing. You are given an array 'ARR' of 0s and 1s, and if there are five consecutive 1s followed by a 0, you have to remove this 0 from the array.

```
"E:\Codeblocks\Error Detection Technique.exe"
Enter 0 or 1 as String: 01011111111101

After Bit Stuffing:
01111110  010111101111100  01111110

After Bit De-Stuffing:
01111110  010111111111100  01111110

Do You Want To Use The Same Program? (y/n):
```

Character Stuffing and De-Stuffing

In byte stuffing (or character stuffing), a particular byte is added to the data section of the frame when there is a character with the same pattern as the flag. The data section is stuffed with an extra byte. This byte is usually known as the escape character (ESC), which has a predefined bit pattern. Whenever the receiver encounters the ESC character, it deletes it from the data section and treats the next character as data, not a delimiting flag.

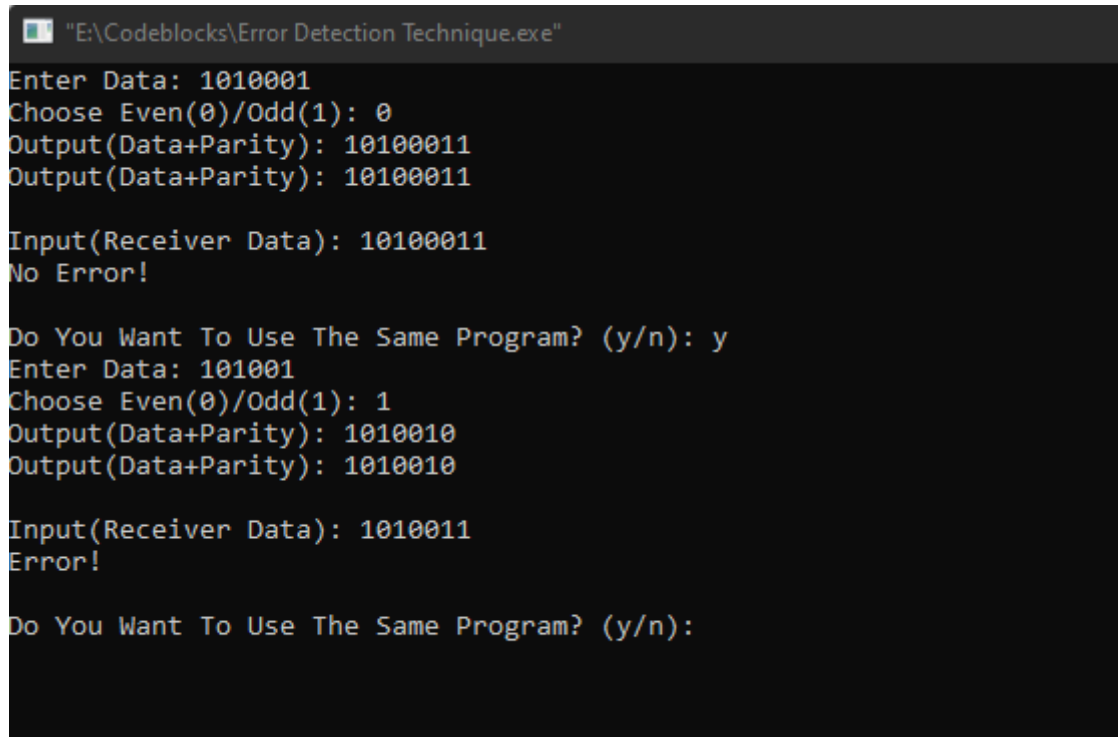
```
"E:\Codeblocks\Error Detection Technique.exe"
Enter String: shanto
Enter starting delimiter : s
Enter ending delimiter : q

Data after stuffing: sqshantos
Data after de stuffing: shanto

Do You Want To Use The Same Program? (y/n):
```

Parity Check

A parity check is an error-correction process in network communication that ensures data transmissions between communication nodes are accurate. In this process, the receiver agrees to use the same even parity bit or odd parity bit scheme as the sender. In an even parity check, parity bits ensure there are an even number of 1s and 0s in the transmission. In an odd parity check, there are an odd number of 1s and 0s in the transmission.



```
"E:\Codeblocks\Error Detection Technique.exe"
Enter Data: 1010001
Choose Even(0)/Odd(1): 0
Output(Data+Parity): 10100011
Output(Data+Parity): 10100011

Input(Receiver Data): 10100011
No Error!

Do You Want To Use The Same Program? (y/n): y
Enter Data: 101001
Choose Even(0)/Odd(1): 1
Output(Data+Parity): 1010010
Output(Data+Parity): 1010010

Input(Receiver Data): 1010011
Error!

Do You Want To Use The Same Program? (y/n):
```

Cyclic Redundancy Check

Cyclic redundancy check (CRC) is a common data transmission error detection technique commonly used in the data communication field. The transmit end calculates a check code for the data in a data frame based on a certain algorithm, appends the check code to the data frame, and sends the data frame to the receive end. The receive end verifies the correctness and integrity of the received data by repeating the calculation using the same algorithm.

```
"E:\Codeblocks\Error Detection Technique.exe"
Enter Sender Message: 100100
Enter Polynomial: 1101
Final Sender Value: 100100001
Enter Receiver Message: 100100001
Enter Polynomial: 1101
No error!

Do You Want To Use The Same Program? (y/n): y
```

Hamming Code

Hamming code is a linear code that is useful for error detection up to two immediate bit errors. It is capable of single-bit errors. In Hamming code, the source encodes the message by adding redundant bits in the message. These redundant bits are mostly inserted and generated at certain positions in the message to accomplish error detection and correction process.

```
"E:\Codeblocks\Error Detection Technique.exe"
Enter 4 Bits of Data One by One
1
0
1
0
Encoded Data is 1010010
Enter Received Data Bits One by One
1 0 1 0 0 1 0
No error!

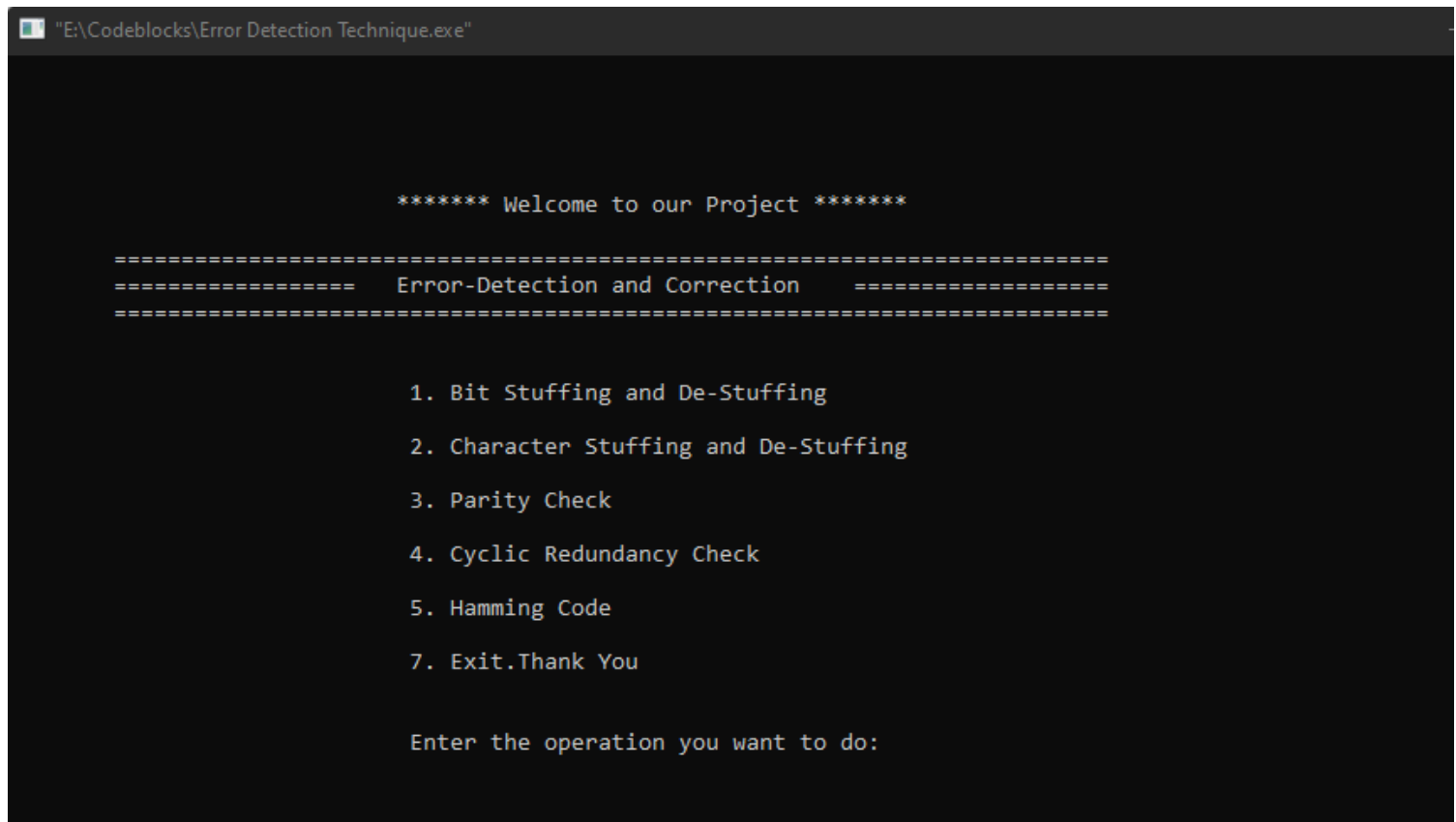
Do You Want To Use The Same Program? (y/n):
```

Chapter 3

Performance Evaluation

3.1 Results and Discussions

From the following we can say that bit stuffing and de-stuffing are used in part of data transmission to avoid error and parity checker, cyclic redundancy checker is used to detect error & hamming code is used for error connection & detection. Here is the final simulation picture.



```
***** Welcome to our Project *****

=====
===== Error-Detection and Correction =====
=====

1. Bit Stuffing and De-Stuffing
2. Character Stuffing and De-Stuffing
3. Parity Check
4. Cyclic Redundancy Check
5. Hamming Code
7. Exit.Thank You

Enter the operation you want to do:
```


Chapter 4

Conclusion

4.1 Introduction

From the following we can say that bit stuffing and de-stuffing are used in part of data transmission to avoid error and parity checker, cyclic redundancy checker is used to detect error & hamming code is used for error connection & detection.

4.1 Scope of Future Work

I will try to add some more error detection & correction technique to this application.
These are -

- Checksum
- Two-dimensional parity checker etc.

References

[1] <https://www.google.com/>

[2] <https://www.youtube.com/>