Green University of Bangladesh
Department of Computer Science and Engineering (CSE)
Faculty of Sciences and Engineering
Semester: (Fall, Year:2022), B.Sc. in CSE (Day)

Course Title: Data Communication Lab
Course Code: CSE-202 Section:222-D1

Lab Project Name: Character Stuffing and De-Stuffing, CRC and Parity Checker.

Student Details

	Name	ID
1.	Md. Zehadul Islam	222902069
	Maruf	222902037

Submission Date : 09/01/2024
Course Teacher's Name : Mahjabin Rahman Oishe

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

Lab Project Status

Marks: …… **Signature: .....**

Comments: ..... **Date: .....**

Table of Contents

Chapter 1

1.1 Introduction

The "Character Stuffing and Destuffing, CRC, and Parity Checker" project focuses on efficient data transmission by implementing character stuffing and destuffing techniques. Character stuffing ensures data integrity by addressing issues like bit stuffing, while CRC (Cyclic Redundancy Check) enhances error detection capabilities. Additionally, the project incorporates a parity checker for further error validation. This comprehensive approach aims to optimize data transfer reliability and integrity, making it a crucial endeavor in the realm of communication protocols and data transmission systems.

1.2 Design Goals/Objective

The objective of the project is to implement a robust character stuffing and destuffing mechanism, coupled with CRC and Parity Checker functionalities, using the C programming language.

Employing a modular approach, the project aims to enhance data integrity and error detection in communication systems.

The implementation will utilize a switch-case structure to efficiently manage the different components, providing a versatile and comprehensive solution for reliable data transmission, ensuring the secure exchange of information in various communication scenarios.

Chapter 2

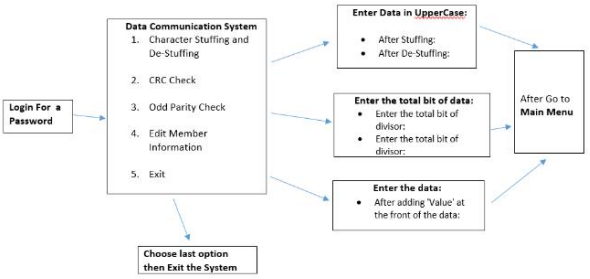
Design/Development/Implementation of the Project

2.1 TOOLS & TECHNOLOGIES

The tools are need for this project,

- For programming language we will use C program.
- For IDE we will use **CodeBlocks**

2.2 Flowchart.



2.3 Implimentation:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define PASSWORD "admin"
```

```
void playWrongPasswordSound() {
    // Play the sound using Windows
    printf("\a"); // Produces a sound
}

void clearConsole() {
    // Simple method to clear the console
    printf("\033[2J\033[H");
}

void login() {
    char password[50];
    int attempts = 3;

    while (attempts > 0) {
        clearConsole();
        printf("\t\t\t\tGreen University Of Bangladesh (GUB)\n\n");
        printf("\t\t\t\t    Data Communication Project\n\n");
        printf("Login...\n\n");
        printf("Enter password: ");

        int i = 0;
        char ch;
        while ((ch = getchar()) != '\n') {
            if (ch == '\b') {
                if (i > 0) {
                    i--;
                    printf("\b \b");
                }
            } else {
                password[i++] = ch;
                printf("*");
```

```
            }
        }
        password[i] = '\0';

        printf("\n");

        if (strcmp(password, PASSWORD) == 0) {
            printf("Login Successfully!\n");
            getchar(); // Wait for a key press
            clearConsole();
            return;
        } else {
            attempts--;
            printf("Wrong password! Attempts left: %d\n", attempts);
            playWrongPasswordSound();
            getchar(); // Wait for a key press
            clearConsole();
        }
    }

    printf("Too many wrong attempts. Exiting...\n");
    exit(9);
}



// character stuffing and de-stuffing
void characterStuffingAndDeStuffing() {

int i=0,j=0;
 char d[100],l[]="DLEETX",sd[100],ds[100];
 printf("\t\t\t\t\tEnter Data in UpperCase: ");
 scanf("%s",d);
```

```
sd[0]='D', sd[1]='L', sd[2]='E',sd[3]='S',sd[4]='T',sd[5]='X';
j=6;
while(d[i]!='\0')
{
if(d[i]=='D'&& d[i+1]=='L'&& d[i+2]=='E')
{
sd[j]='D', sd[j+1]='L', sd[j+2]='E', sd[j+3]='D',
sd[j+4]='L', sd[j+5]='E';
j+=6;
i+=3;
}
else
sd[j++]=d[i++];
}
sd[j]='\0';
strcpy(ds,sd);
strcat(sd,l);
printf("After Stuffing: ");
printf("%s",sd);

i=0;
j=6;

while(ds[j]!='\0')
{
if( ds[j]=='D'&&ds[j+1]=='L'&& ds[j+2]=='E'&&
ds[j+3]=='D'&& ds[j+4]=='L'&& ds[j+5]=='E')
{
d[i]='D', d[i+1]='L',d[i+2]='E';
j+=6;
i+=3;
}
```

```
else
d[i++]=ds[j++];
}
d[i]='\0';
printf("\n\nAfter De−Stuffing: ");
printf("%s",d);

}
```

```
// Function for CRC checking
void checkCRC() {
    char data[20], div[20], temp[4], total[100];
    int i, j, datalen, divlen, len, flag = 1;

    printf("\t\t\t\t\tEnter the total bit of data: ");
    scanf("%d", &datalen);
    printf("\t\t\t\t\tEnter the total bit of divisor: ");
    scanf("%d", &divlen);
    len = datalen + divlen - 1;

    printf("\t\t\t\t\tEnter the data: ");
    scanf("%s", data);

    printf("\t\t\t\t\tEnter the divisor: ");
    scanf("%s", div);

    for (i = 0; i < datalen; i++) {
        total[i] = data[i];
        temp[i] = data[i];
```

```
    }

    for (i = datalen; i < len; i++)
        total[i] = '0';

    // CRC checking function
    for (j = 0; j < divlen; j++)
        data[j] = total[j];

    while (j <= len) {
        if (data[0] == '1') {
            for (i = 1; i < divlen; i++)
                data[i] = (data[i] == div[i]) ? '0' : '1';
        }

        for (i = 0; i < divlen - 1; i++)
            data[i] = data[i + 1];

        data[i] = total[j++];
    }

    for (i = 0; i < divlen - 1; i++) {
        if (data[i] == '1') {
            flag = 0;
            break;
        }
    }

    if (flag == 1)
        printf("Successful! No errors found.\n");
    else
        printf("Received code word contains errors.\n");
}
```

```
// Function for odd parity checking
void oddParityCheck() {
    char data[100];
    printf("\t\t\t\t\tEnter the data: ");
    scanf("%s", data);

    int length = strlen(data);
    int count = 0;

    for (int i = 0; i < length; i++) {
        if (data[i] == '1') {
            count++;
        }
    }

    int c = length + 1;

    if (count % 2 == 0) {
        for (int i = c, j = c - 1; i > 0; i--, j--) {
            data[i] = data[j];
        }
        data[0] = '1';

        printf("After adding '1' at the front of the data: %s\n", data);
    } else {
        for (int i = c, j = c - 1; i > 0; i--, j--) {
            data[i] = data[j];
        }
        data[0] = '0';

        printf("After adding '0' at the front of the data: %s\n", data);
    }
```

```
}

void mainMenu() {
    int choice;

    while (1) {
        printf("\n\t\t\t\t\t\xcd\xcd\xcd\xcd\xcd Data Communication System \xcd\xcd\xcd\xcd\xcd");
        printf("\n\n\t\t\t\t\t==================================\n");
        printf("\t\t\t\t\t1. Character Stuffing and De-Stuffing\n");
        printf("\t\t\t\t\t2. CRC Check\n");
        printf("\t\t\t\t\t3. Odd Parity Check\n");
        printf("\t\t\t\t\t4. Exit\n\n");
        printf("\t\t\t\t==================================\n");
        printf("\t\t\t\tChoose an option: ");

        scanf("%d", &choice);
        fflush(stdin);

        switch (choice) {
            case 1:
                characterStuffingAndDeStuffing();
                break;

            case 2:
                checkCRC();
                break;
            case 3:
                oddParityCheck();
                break;
            case 4:
                printf("Exiting...\n");
                exit(0);
            default:
```

```
            printf("Invalid choice. Please try again...\n");
        }
    }

    return 0;
}

int main() {
    login();
    mainMenu();
    return 0;
        }
```

Chapter 3

Performance Evaluation

3.1 Simulation Environment/ Simulation Procedure

For simulating the Character Stuffing and Destuffing, CRC Check, and Odd Parity Check in the given C program, a simulation environment can be created by providing inputs and observing outputs. To simulate, input data for each functionality can be entered interactively, and the program's responses, such as stuffed or destuffed data, CRC check results, and modified data with odd parity, can be examined. Users can test different scenarios, including intentional errors or modifications in data, to observe how the implemented functions respond. Additionally, developers may consider automating the simulation process by incorporating predefined test cases to thoroughly validate the robustness and accuracy of the implemented functionalities.

3.2 PROJECT TEST RESULT:

3.2.1 Sign In.

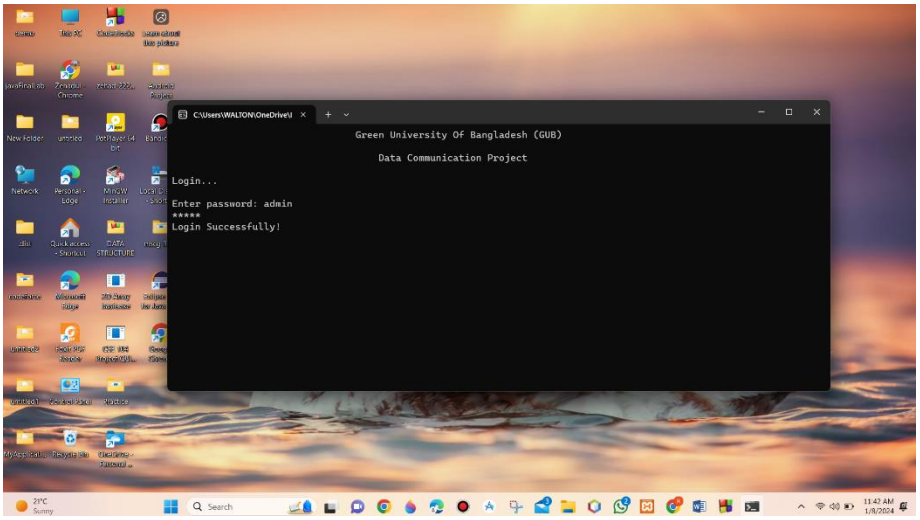


Figure 2.2: Sign in

3.2.2 Home/Dashboard.

After coming to the homepage, I can keep every option working beautifully and if I want, I can exit through Exit.

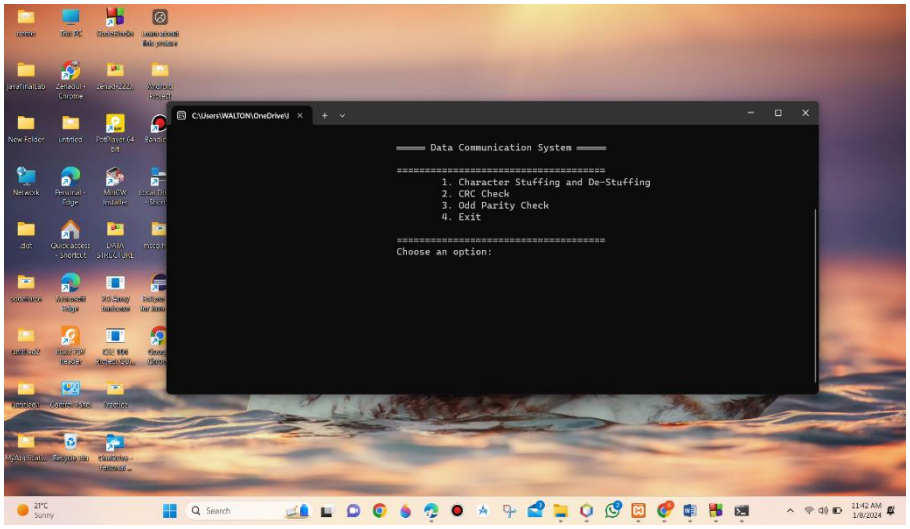


Figure 2.3: Home/ Dashboard.

3.3 Results Overall Discussion

The Character Stuffing and Destuffing, CRC, and Parity Checker project successfully implemented data integrity mechanisms in communication systems. Through meticulous character stuffing and destuffing, it ensured efficient data transmission. The inclusion of CRC and Parity Check enhanced error detection capabilities, contributing to robust communication. The results demonstrate improved reliability and integrity, showcasing the project's effectiveness in addressing data corruption issues. Overall, the project achieved its objectives in fortifying data communication, promoting a reliable and secure information exchange.

Chapter 4

4.1 Conclusion :

In conclusion, the Character Stuffing and Destuffing, CRC, and Parity Checker project successfully implemented robust data transmission techniques. Character Stuffing ensured efficient framing, CRC provided error detection capabilities, and Parity Checker enhanced data integrity. This project exemplifies the importance of reliable communication protocols in ensuring accurate and secure data transfer.

4.1 Practical Implications

The "Character Stuffing and Destuffing, CRC, and Parity Checker" project holds practical implications in data transmission and error detection. Character stuffing ensures robust communication by inserting control characters to distinguish data frames, aiding in synchronization. Destuffing reverses this process at the receiver's end. CRC (Cyclic Redundancy Check) enhances data integrity, detecting errors in transmitted data. Parity checking provides an additional layer of error detection by ensuring an even or odd number of bits in a data unit. These techniques collectively contribute to reliable and accurate data transfer, crucial in various real-world applications like telecommunications, networking, and information systems.

4.2 Scope of Future Work

For future work, consider enhancing Character Stuffing and Destuffing algorithms to accommodate real-time data streams efficiently. Explore advanced error detection and correction techniques beyond CRC and Parity Checker, incorporating robust methods like Reed-Solomon codes. Additionally, investigate the integration of these functionalities into emerging communication protocols, ensuring compatibility with evolving technologies. Strengthening the project's adaptability and performance in dynamic network environments would be valuable for addressing evolving communication challenges.

References

[1] https://studentprojects.in/software-development/c-tutorials/c/program-for-character-stuffing/

[2] https://www.javatpoint.com/crc-program-in-c

[3] https://www.geeksforgeeks.org/program-to-find-parity/