



Green University of Bangladesh

Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering

Semester: (Spring, Year:2025), B.Sc. in CSE (Day)

LAB REPORT NO - 01

Course Title: Data Communication

Course Code: CSE307 Section:223-D1

**Lab Experiment Name : Implementing Byte (Character) Stuffing
and De-stuffing**

Student Details

Name		ID
1.	MD.SHAJALAL	223002088

Lab Date : 24 - 02 - 2025

Submission Date : 03 – 03 - 2025

Course Teacher's Name : Md.Samin Hossain Utsho

Lab Report Status

Marks:

Comments:.....

Signature:.....

Date:.....

1. TITLE OF THE LAB REPORT EXPERIMENT

Implementing Byte (Character) Stuffing and De-stuffing

2. Objective :

The objective of this experiment is to implement Byte (Character) Stuffing and De-Stuffing in a programming language (C) and demonstrate how the technique ensures reliable data transmission.

3. Byte Stuffing Algorithm:

1. Define a **flag** character that marks the start and end of a frame.
2. Define an **escape** character to be used for stuffing.
3. Traverse the data string:
 - If a character matches the flag or escape character, prepend it with the escape character.
 - Otherwise, append it normally.
4. Add the flag at the beginning and end of the frame.
5. Return the stuffed data.

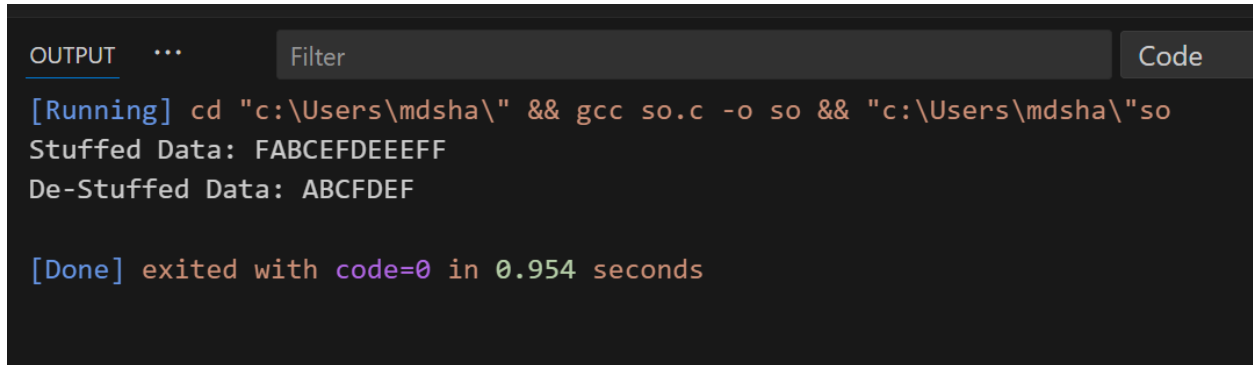
4. Byte De-Stuffing Algorithm:

1. Ensure the received frame starts and ends with the flag character.
2. Remove the flag characters from the start and end.
3. Traverse the data:
 - If an escape character is encountered, skip it and append the next character.
 - Otherwise, append the character normally.
4. Return the de-stuffed data.

5. Implementation (C Code) :

```
6. #include <stdio.h>
7. #include <string.h>
8. #define FLAG 'F'
9. #define ESC 'E'
10.
11. void byte_stuffing(char *data, char *stuffed) {
12.     int j = 0;
13.     stuffed[j++] = FLAG;
14.     for (int i = 0; i < strlen(data); i++) {
15.         if (data[i] == FLAG || data[i] == ESC) {
16.             stuffed[j++] = ESC;
17.         }
18.         stuffed[j++] = data[i];
19.     }
20.     stuffed[j++] = FLAG;
21.     stuffed[j] = '\0';
22. }
23.
24. void byte_de_stuffing(char *stuffed, char *destuffed) {
25.     int j = 0;
26.     for (int i = 1; i < strlen(stuffed) - 1; i++) {
27.         if (stuffed[i] == ESC) {
28.             i++;
29.         }
30.         destuffed[j++] = stuffed[i];
31.     }
32.     destuffed[j] = '\0';
33. }
34.
35. int main() {
36.     char data[] = "ABCFDEF";
37.     char stuffed[50], destuffed[50];
38.
39.     byte_stuffing(data, stuffed);
40.     printf("Stuffed Data: %s\n", stuffed);
41.
42.     byte_de_stuffing(stuffed, destuffed);
43.     printf("De-Stuffed Data: %s\n", destuffed);
44.
45.     return 0;
46. }
```

6. Program Output:

A screenshot of a terminal window with a dark background. At the top, there is a header bar with the word 'OUTPUT' on the left, a 'Filter' input field in the center, and a 'Code' button on the right. Below the header, the terminal displays the following text: '[Running] cd "c:\Users\mdsha\" && gcc so.c -o so && "c:\Users\mdsha\"so', 'Stuffed Data: FABCEFDEEEFF', 'De-Stuffed Data: ABCFDEF', and '[Done] exited with code=0 in 0.954 seconds'.

```
OUTPUT  ...  Filter  Code
[Running] cd "c:\Users\mdsha\" && gcc so.c -o so && "c:\Users\mdsha\"so
Stuffed Data: FABCEFDEEEFF
De-Stuffed Data: ABCFDEF

[Done] exited with code=0 in 0.954 seconds
```

6. Conclusion : Byte Stuffing and De-Stuffing are essential techniques in data transmission to handle special control characters within a message. This experiment successfully implemented both stuffing and de-stuffing in C, ensuring that data integrity is maintained during transmission. The output confirms the correctness of the algorithm.