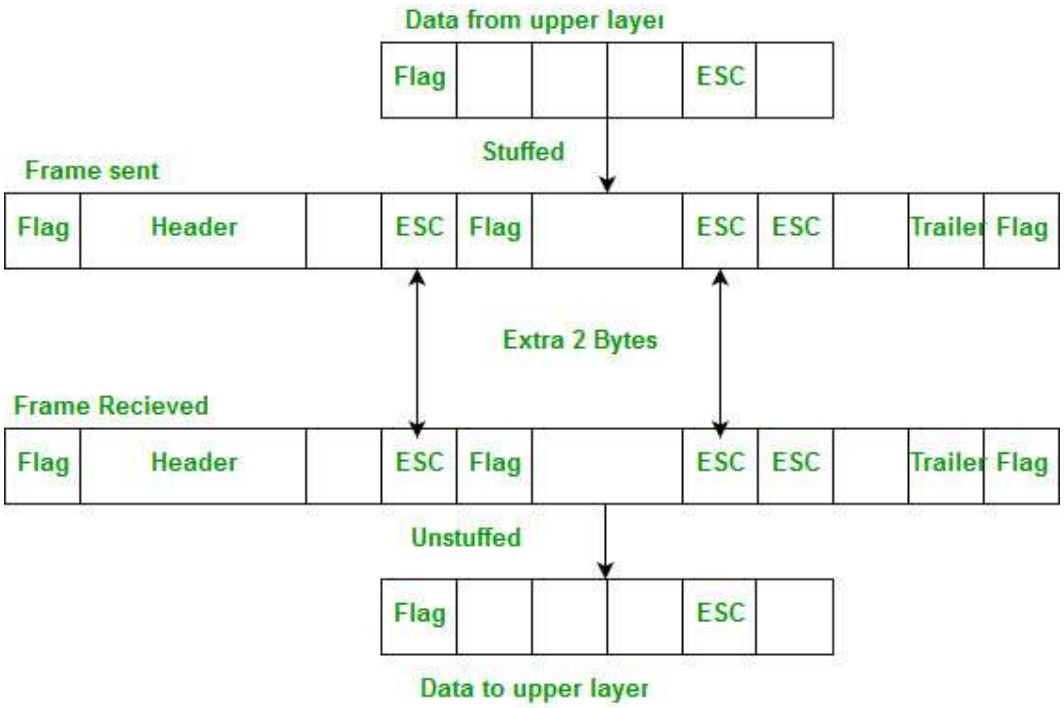
PRACTICAL - 6

DATE: , Wednesday

AIM: Write a program which demonstrates the concept of byte stuffing.

- **What is Byte-Stuffing?**
  - Byte stuffing is a data transmission technique used in computer networks to prevent special control characters within the data from being mistaken as frame delimiters or other control signals.
  - It involves inserting an escape character before any byte that matches a delimiter or the escape character itself.
  - For example, if a frame delimiter is 0x7E and an escape character is 0x7D, then any occurrence of 0x7E in the data is replaced with 0x7D 0x5E, and any 0x7D is replaced with 0x7D 0x5D.
  - This ensures the data is transmitted correctly without being misinterpreted by the receiving system. Byte stuffing is commonly used in protocols like HDLC and PPP to maintain data integrity in network communications.



Code:

```
#include <iostream>
#include <string>
#include <thread>
#include <cstring>
#include <unistd.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#define PORT 45678
using namespace std;
string byteStuffing(const string& data) {
    string stuffed_data = "F" + data + "F";
    string res;
    for (char ch : stuffed_data) {
        if ((ch == 'F' || ch == 'E') && ch != stuffed_data.front() && ch != stuffed_data.back()) {
            res += 'E';
        }
        res += ch;
    }
    return res;
}
string byteDestuffing(const string& data) {
    string destuffed_data;

    for (size_t i = 1; i < data.length() - 1; ++i) {
        if (data[i] == 'E' && (data[i + 1] == 'F' || data[i + 1] == 'E')) {
            destuffed_data += data[i + 1];
            ++i;
        } else {
            destuffed_data += data[i];
        }
    }
    return destuffed_data;
}
void server() {
    int server_fd, new_socket;
    struct sockaddr_in address;
    int opt = 1;
    int addrlen = sizeof(address);

    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
        perror("Socket failed");
        exit(EXIT_FAILURE);
    }
    if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt,
sizeof(opt))) {
        perror("Setsockopt failed");
        close(server_fd);
        exit(EXIT_FAILURE);
    }
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(PORT);
```

```
    if (bind(server_fd, (struct sockaddr*)&address, sizeof(address)) < 0) {
        perror("Bind failed");
        close(server_fd);
        exit(EXIT_FAILURE);
    }
    if (listen(server_fd, 3) < 0) {
        perror("Listen failed");
        close(server_fd);
        exit(EXIT_FAILURE);
    }
    cout << "Server listening on port " << PORT << "..." << endl;
    if ((new_socket = accept(server_fd, (struct sockaddr*)&address, (socklen_t*)&addrlen)) < 0) {
        perror("Accept failed");
        close(server_fd);
        exit(EXIT_FAILURE);
    }
    char buffer[1024] = {0};
    int valread = read(new_socket, buffer, 1024);
    string received_data(buffer, valread);
    cout << "Message Received... Successfully!!!" << endl;
    cout << "The Stuffed Message is: " << received_data << endl;
    string destuffed_data = byteDestuffing(received_data);
    cout << "The Destuffed Message is: " << destuffed_data << endl;
    send(new_socket, "success", strlen("success"), 0);
    valread = read(new_socket, buffer, 1024);
    string exit_message(buffer, valread);
    if (exit_message == "bye") {
        cout << "Messaging is over... EXITING" << endl;
    }
    close(new_socket);
    close(server_fd);
}
void client() {
    sleep(1);
    int sock = 0;
    struct sockaddr_in serv_addr;

    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        cout << "Socket creation error" << endl;
        return;
    }
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);

    if (inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr) <= 0) {
        cout << "Invalid address/ Address not supported" << endl;
        return;
    }
    if (connect(sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr)) < 0) {
        cout << "Connection Failed" << endl;
        return;
    }
    string data;
    cout << "Enter the Message to be Sent: ";
    getline(cin, data);
```

```
    string stuffed_data = byteStuffing(data);
    cout << "The data being sent (with byte stuffing) is: " << stuffed_data << endl;
    send(sock, stuffed_data.c_str(), stuffed_data.size(), 0);
    cout << "Sending Message..." << endl;
    char buffer[1024] = {0};
    int valread = read(sock, buffer, 1024);
    if (string(buffer, valread) == "success") {
        cout << "Thanks for the Feedback Server!!" << endl;
    }
    send(sock, "bye", strlen("bye"), 0);
    close(sock);
}
int main() {
    thread server_thread(server);
    client();
    server_thread.join();
    return 0;
}
```

Output:

```
Server listening on port 45678...
Enter the Message to be Sent: DFEDDFED
The data being sent (with byte stuffing) is: FDFEEDDFEEDF
Sending Message...Message Received... Successfully!!!
The Stuffed Message is: FDFEEDDFEEDF
The Destuffed Message is: DFEDDFED

Thanks for the Feedback Server!!
Messaging is over... EXITING
```

Date of Submission: **Sign:**

_____ **Mr. Jigar Patel**