Experiment – 4

AIM- Write a program to implement data link layer stuffing methods such as.

- Bit Stuffing
- Byte Stuffing
- Character Stuffing

Description - Data link layer is responsible for something called Framing, which is the division of stream of bits from network layer into manageable units (called frames). Each frame consists of sender's address and a destination address.The destination address defines where the packet is to go and the sender's address helps the recipient acknowledge the receipt. Frames could be of fixed size or variable size. In fixed-size framing, there is no need for defining the boundaries of the frames as the size itself can be used to define the end of the frame and the beginning of the next frame. But, in variable-size framing, we need a way to define the end of the frame and the beginning of the next frame. To separate one frame from the next, an 8-bit (or 1-byte) flag is added at the beginning and the end of a frame. But the problem with that is, any pattern used for the flag could also be part of the information. So, there are two ways to overcome this problem:

- Using Byte stuffing
- Using Character stuffing
- Using Bit stuffing

Algorithms:

Bit-Stuffing:

1. Start.
2. Initialise the array with the special bit pattern 0111 1110 which indicates the beginning ofthe frame.
3. Get the bit string to be transmitted in the array.
4. Check for five consecutive 1's and if they occur ,stuff a bit 0.
5. Display the data transmitted as it appear on the dateline after appending 01111110 at the end.
6. For stuffing copy the transmitted data to another array after detecting the stuffed bits.
7. Display the received bit stream.
8. Stop.

Byte-Stuffing:

1. Start.
2. Append 01111110 at the beginning of the string.
3. Check the data if character is present ; if character 01111110 is present in the string(example 0101111110101) insert another 01111110 in the string.
4. Transmit 01111110 at the end of the string.
5. Display the string.
6. Stop.

Character-Stuffing:

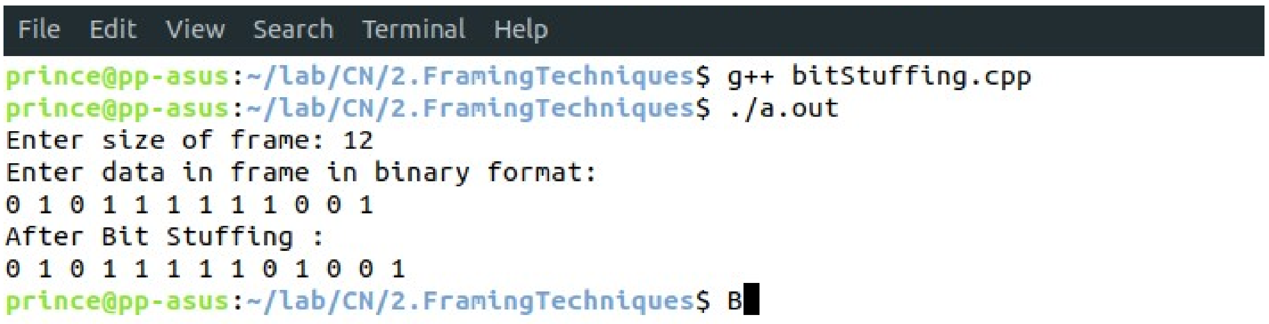
1. Start.
2. Append DLE STX at the beginning of the string.
3. Check the data if character is present ; if character DLE is present in the string insert another DLE in the string.
4. Transmit DLE ETX at the end of the string.
5. Display the string.
6. Stop.

BIT-STUFFING:

Code -

```
#include <iostream>
using namespace std;

int main(){
    int a[20], b[30], k, n,i,j;

    cout<<"Enter size of frame: ";
    cin>>n;
    cout<<"Enter data in frame in binary format: \n";
    for (i = 0; i < n; i++){
        cin>>a[i];
    }

    int c = 1;
    for (i=0,j=0;i < n;i++,j++){
        if (a[i] == 1){
            b[j] = a[i];
            for (k = i + 1; (a[k] == 1) && (k < n) && (c < 5); k++){
                b[++j] = a[k];
                c++;
                if (c == 5){
                    b[++j] = 0;
                }
                i = k;
            }
        }else{
            b[j] = a[i];
        }
    }

    cout<<"After Bit Stuffing :\n";
    for (i = 0; i < j; i++){
        cout<<b[i]<<" ";
    }
    cout<<endl;

    return 0;
}
```

Output –



```
prince@pp-asus:~/lab/CN/2.FramingTechniques$ g++ bitStuffing.cpp
prince@pp-asus:~/lab/CN/2.FramingTechniques$ ./a.out
Enter size of frame: 12
Enter data in frame in binary format:
0 1 0 1 1 1 1 1 1 0 0 1
After Bit Stuffing :
0 1 0 1 1 1 1 1 0 1 0 0 1
prince@pp-asus:~/lab/CN/2.FramingTechniques$ B
```

BYTE-STUFFING:

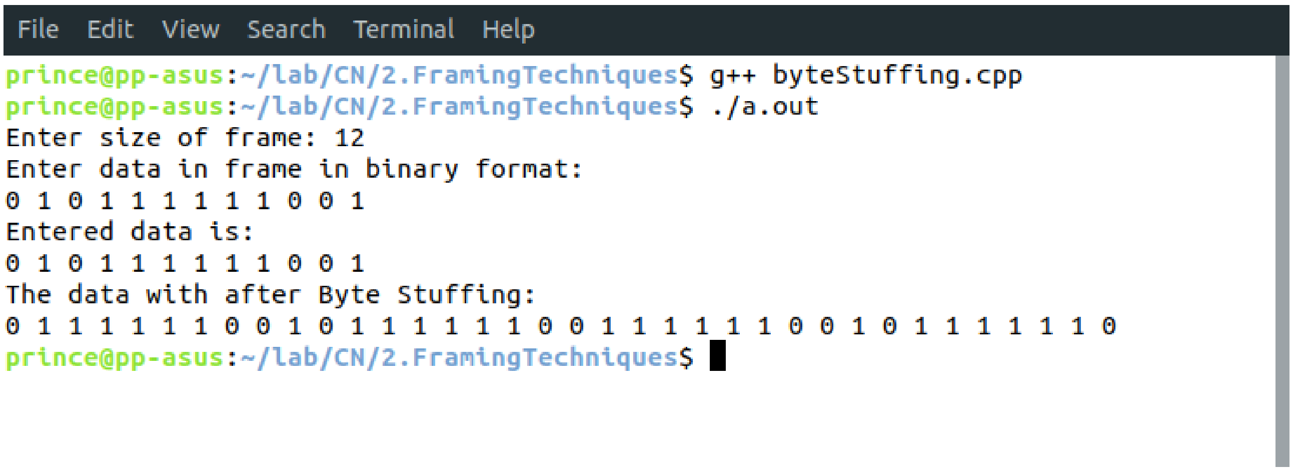
Code-

```
#include <iostream>

using namespace std;

int main(){
    int i, j, k, data[100], n;

    cout<<"Enter size of frame: ";
    cin>>n;
```

```
cout<<"Enter data in frame in binary format: "<<endl;
for (i = 0; i < n; i++){
    cin>>data[i];
}

cout<<"Entered data is: "<<endl;
for (int i = 0; i < n; i++){
    cout<<data[i]<<" ";
}
cout<<endl;

for (int i = 0; i < n; i++){
    if (data[i] == 0 &&
       data[i + 1] == 1 && data[i + 2] == 1 &&
        data[i + 3] == 1 && data[i + 4] == 1 &&
        data[i + 5] == 1 && data[i + 6] == 1 &&
       data[i + 7] == 0){

        i = i + 8;
        for (j = i + 8; j >= i; j--){
            data[j + 8] = data[j];
            n++;
        }

        for (int k = 0; k < 8; k++){
            if (k == 0 || k == 7){
                data[i] = 0;
            }
            else{
                data[i] = 1;
            }
            i++;
        }
    }
}

cout<<"The data with after Byte Stuffing: "<<endl;
```

```
    cout<<"0 1 1 1 1 1 1 0 ";
    for (int i = 0; i < n - 1; i++)
    {
        cout<<data[i]<<" ";
    }
    cout<<"0 1 1 1 1 1 1 0"<<endl;

    return 0;
}
```

Output -



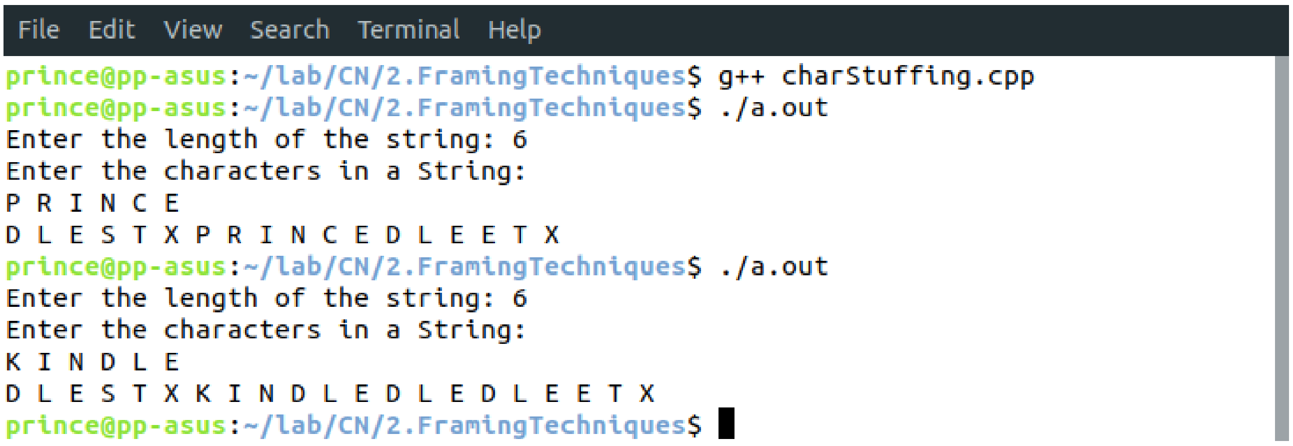
```
prince@pp-asus:~/lab/CN/2.FramingTechniques$ g++ byteStuffing.cpp
prince@pp-asus:~/lab/CN/2.FramingTechniques$ ./a.out
Enter size of frame: 12
Enter data in frame in binary format:
0 1 0 1 1 1 1 1 1 0 0 1
Entered data is:
0 1 0 1 1 1 1 1 1 0 0 1
The data with after Byte Stuffing:
0 1 1 1 1 1 1 0 0 1 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 0 1 0 1 1 1 1 1 1 0
prince@pp-asus:~/lab/CN/2.FramingTechniques$
```

CHARACTER-STUFFING:

Code -

```
#include <iostream>
using namespace std;

int main(){
    char s[50], data[100];
    int m,n;

    cout<<"Enter the length of the string: ";
    cin>>n;
    cout<<"Enter the characters in a String: "<<endl;
    for (int i = 0; i < n; i++){
        cin>>s[i];
    }
    data[0] = 'D';
    data[1] = 'L';
    data[2] = 'E';
    data[3] = 'S';
    data[4] = 'T';
    data[5] = 'X';
    m = 6;
    for (int i=0, j=6; i < n; i++){
        if (s[i] == 'D' &&
            s[i + 1] == 'L' &&
            s[i + 2] == 'E'){

            data[j++] = 'D';
            data[j++] = 'L';
            data[j++] = 'E';
            data[j++] = 'D';
            data[j++] = 'L';
            data[j++] = 'E';
            i += 3;
            m += 6;
        }else{
            data[j++] = s[i];
            m++;
        }
    }
    data[m++] = 'D';
    data[m++] = 'L';
    data[m++] = 'E';
    data[m++] = 'E';
    data[m++] = 'T';
    data[m++] = 'X';

    for (int i = 0; i < m; i++){
        cout<<data[i]<<" ";
    }
    cout<<endl;

    return 0;
}
```

Output:



```
prince@pp-asus:~/lab/CN/2.FramingTechniques$ g++ charStuffing.cpp
prince@pp-asus:~/lab/CN/2.FramingTechniques$ ./a.out
Enter the length of the string: 6
Enter the characters in a String:
P R I N C E
D L E S T X P R I N C E D L E E T X
prince@pp-asus:~/lab/CN/2.FramingTechniques$ ./a.out
Enter the length of the string: 6
Enter the characters in a String:
K I N D L E
D L E S T X K I N D L E D L E D L E E T X
prince@pp-asus:~/lab/CN/2.FramingTechniques$ 
```

Learnings – FINDINGS AND LEARNINGS :

1) Whenever a message is transmitted, it may get scrambled by noise or data may get corrupted. To avoid this, we use error-detecting codes which are additional data added to a given digital message to help us detect if any error has occurred during transmission of the message.

2) Unlike other schemes, which are based on addition, CRC is based on binary division.

3) CRCs are specifically designed to protect against common types of errors on communication channels, where they can provide quick and reasonable assurance of the integrity of messages delivered.

4) The device may take corrective action, such as rereading the block or requesting that it be sent again. Otherwise, the data is assumed to be error-free (though, with some small probability, it may contain undetected errors; this is the fundamental nature of error-checking).