EXPERIMENT 4

OBJECTIVE:

C PROGRAM TO IMPLEMENT CHARACTER COUNT IN DATA LINK LAYER

Data link layer translates the physical layers raw bit stream into discrete messages called frames. First framing method uses a field in the header to specify the number of characters in the frame. When the data link layer at the destination sees the character count, it knows how many characters follow and hence where the end of the frame is.

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

// Function to calculate character count
intcalculate_character_count(char *data) {
returnstrlen(data);
}

// Function to frame data with character count
voidframe_data(char *data, char *framed_data) {
intchar_count = calculate_character_count(data);
sprintf(framed_data, "%d%s", char_count, data);
}

// Function to unframe data and verify character count
voidunframe_data(char *framed_data, char *unframed_data) {
intchar_count = atoi(framed_data);
if (strlen(framed_data + 1) == char_count) {
strcpy(unframed_data, framed_data + 1);
printf("Data received correctly\n");
    } else {
printf("Error in data transmission\n");
    }
}
int main() {
char data[] = "Hello, World!";
charframed_data[256];
charunframed_data[256];
printf("Original Data: %s\n", data);

    // Frame data with character count
frame_data(data, framed_data);
printf("Framed Data: %s\n", framed_data);

    // Unframe data and verify character count
unframe_data(framed_data, unframed_data);
printf("Unframed Data: %s\n", unframed_data);
return 0;
}
```

Output

2. Bit Stuffing in Data Link Layer using C

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.

The idea is to check if the given array consists of 5 consecutive **1's**.

Follow the steps below to solve the problem:

- Initialize the array brr[] which stores the stuffed array. Also, create a variable count which maintains the count of the consecutive 1's.
- Traverse in a while loop using a variable i in the range [0, N) and perform the following tasks:
  - If **arr[i]** is **1** then check for the next **4** bits if they are set bits as well. If they are, then insert a 0 bit after inserting all the 5 set bits into the array **brr[]**.
  - Otherwise, insert the value of **arr[i]** into the array **brr[]**.

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

// Function to perform bit stuffing
voidbitStuffing(const char* input, char* output) {
int i = 0, j = 0, count = 0;
intlen = strlen(input);

while (i <len) {
output[j++] = input[i];
if (input[i] == '1') {
count++;
if (count == 5) {
            // Insert a '0' after five consecutive '1's
output[j++] = '0';
count = 0;
          }
      } else {
count = 0;
      }
```

```
i++;
   }
output[j] = '\0';
}

// Function to perform bit destuffing
voidbitDestuffing(const char* input, char* output) {
int i = 0, j = 0, count = 0;
intlen = strlen(input);

while (i <len) {
output[j++] = input[i];
if (input[i] == '1') {
count++;
if (count == 5) {
            // Skip the stuffed '0'
if (input[i+1] == '0') {
i++;
            }
count = 0;
         }
      } else {
count = 0;
      }
i++;
   }
output[j] = '\0';
}

int main() {
const char* input = "011111101111110";
charstuffedOutput[256];
chardestuffedOutput[256];

printf("Original Input: %s\n", input);

bitStuffing(input, stuffedOutput);
printf("Bit Stuffed Output: %s\n", stuffedOutput);

bitDestuffing(stuffedOutput, destuffedOutput);
printf("Bit Destuffed Output: %s\n", destuffedOutput);

return 0;
}
```

```
Original Input: 011111101111110
Bit Stuffed Output: 01111101011111010
Bit Destuffed Output: 011111101111110
```