

```

// Server details

server_addr.sin_family = AF_INET;

server_addr.sin_port = htons(PORT);

if (inet_pton(AF_INET, SERVER_IP, &server_addr.sin_addr) <= 0) {
    perror("Invalid address");
    exit(EXIT_FAILURE);
}

// Connect

if (connect(sock, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
    perror("Connection failed");
    exit(EXIT_FAILURE);
}

printf("Connected to chat server. Start chatting...\n");

// Chat loop

while (1) {
    // Send message

    printf("Client: ");

    fgets(buffer, BUFFER_SIZE, stdin);

    send(sock, buffer, strlen(buffer), 0);

    if (strncmp(buffer, "exit", 4) == 0)
        break;

    // Receive message

    int bytes_received = read(sock, buffer, BUFFER_SIZE - 1);

    if (bytes_received <= 0) {
        printf("Server disconnected.\n");
        break;
    }

    buffer[bytes_received] = '\0';

    printf("Server: %s", buffer);

    if (strncmp(buffer, "exit", 4) == 0)
        break;
}

close(sock);

```

```
    return 0;
}
```

Output:

**Server**

Chat server running on port 8080...

Client connected. Start chatting...

Client: Hello!

Server: Hi there!

Client: exit

**Client**

Connected to chat server. Start chatting...

Client: Hello!

Server: Hi there!

Client: exit

**Ex:No : 30**

**Implementation of Bit stuffing mechanism using C**

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

// Function to perform bit stuffing
void bitStuffing(int arr[ ], int n)
{
    int stuffed[100];

    int i, j = 0, count = 0;

    for (i = 0; i < n; i++)
    {
        stuffed[j] = arr[i];
        if (arr[i] == 1) {
            count++;
            if (count == 5) {
                j++;
                stuffed[j] = 0; // Insert a 0 after five consecutive 1's
                count = 0;
            }
        }
    }
    else
    {
        count = 0;
    }

    j++;
}

printf("Data after Bit Stuffing: ");
for (i = 0; i < j; i++)
{
```

```

        printf("%d", stuffed[i]);
    }
    printf("\n");
}

// Function to perform bit de-stuffing
void bitDeStuffing(int arr[ ], int n)
{
    int destuffed[100];
    int i, j = 0, count = 0;

    for (i = 0; i < n; i++)
    {
        destuffed[j] = arr[i];
        if (arr[i] == 1) {
            count++;
            if (count == 5)
            {
                i++; // Skip the stuffed zero
                count = 0;
            }
        }
    }
    else {
        count = 0;
    }
    j++;
}

printf("Data after Bit De-stuffing: ");
for (i = 0; i < j; i++) {
    printf("%d", destuffed[i]);
}
printf("\n");
}

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

    printf("Enter number of bits: ");
    scanf("%d", &n);

    printf("Enter the bits (0/1):\n");
    for (i = 0; i < n; i++) {
        scanf("%d", &data[i]);
    }

    bitStuffing(data, n);

    printf("Enter stuffed data length: ");
    int stuffed_len;
    scanf("%d", &stuffed_len);
    int stuffed_data[100];
    printf("Enter the stuffed bits (0/1):\n");
    for (i = 0; i < stuffed_len; i++) {
        scanf("%d", &stuffed_data[i]);
    }

    bitDeStuffing(stuffed_data, stuffed_len);
}

```

```
    return 0;
}
```

### Output:

Enter number of bits: 8

Enter the bits (0/1):

0 1 1 1 1 1 0

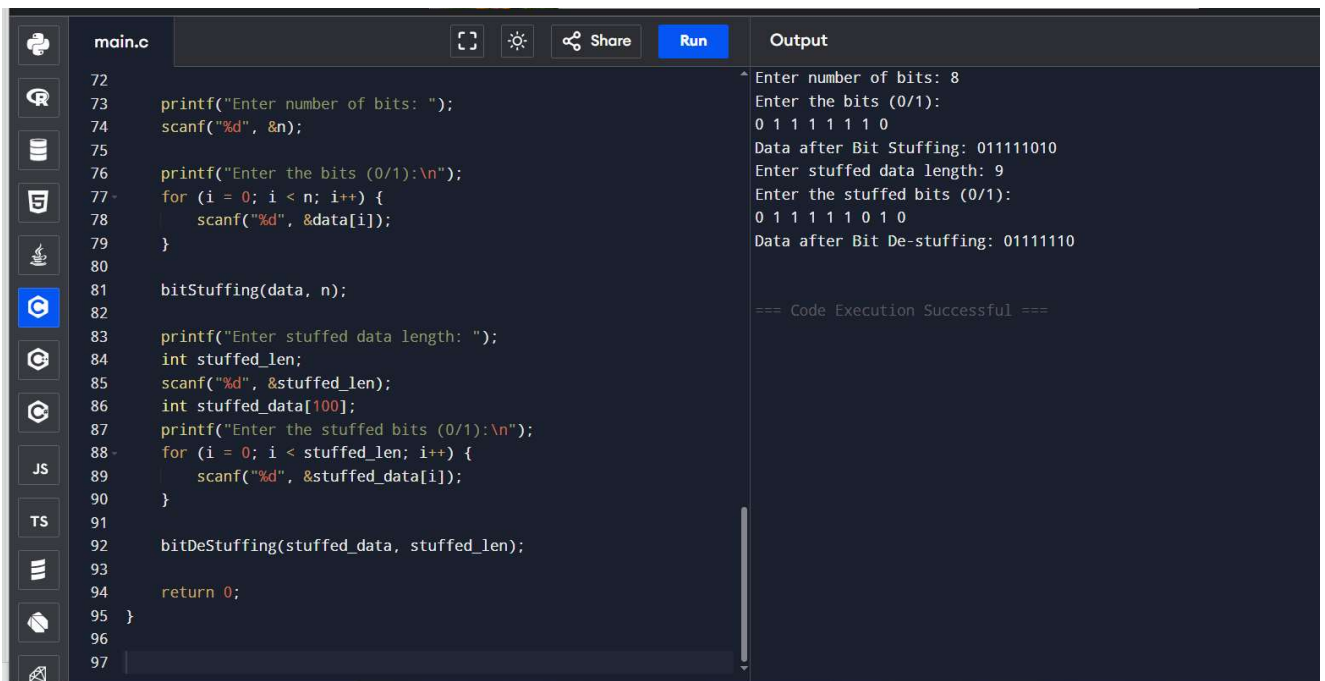
Data after Bit Stuffing: 011111010

Enter stuffed data length: 9

Enter the stuffed bits (0/1):

0 1 1 1 1 1 0 1 0

Data after Bit De-stuffing: 01111110



The screenshot shows a C++ IDE with a file named `main.c`. The code implements a bit stuffing algorithm. It prompts the user for the number of bits (8), the bits themselves (01111110), the stuffed data length (9), and the stuffed bits (0111111010). It then performs bit de-stuffing and outputs the original data (01111110). The output window shows the same sequence of prompts and results, followed by a success message: `=== Code Execution Successful ===`.

```
main.c
72
73     printf("Enter number of bits: ");
74     scanf("%d", &n);
75
76     printf("Enter the bits (0/1):\n");
77     for (i = 0; i < n; i++) {
78         scanf("%d", &data[i]);
79     }
80
81     bitStuffing(data, n);
82
83     printf("Enter stuffed data length: ");
84     int stuffed_len;
85     scanf("%d", &stuffed_len);
86     int stuffed_data[100];
87     printf("Enter the stuffed bits (0/1):\n");
88     for (i = 0; i < stuffed_len; i++) {
89         scanf("%d", &stuffed_data[i]);
90     }
91
92     bitDeStuffing(stuffed_data, stuffed_len);
93
94     return 0;
95 }
96
97
```

Output

```
Enter number of bits: 8
Enter the bits (0/1):
0 1 1 1 1 1 0
Data after Bit Stuffing: 011111010
Enter stuffed data length: 9
Enter the stuffed bits (0/1):
0 1 1 1 1 1 0 1 0
Data after Bit De-stuffing: 01111110

=== Code Execution Successful ===
```