Echo from server: How are you?

Server

TCP Echo Server running on port 8080...

Client connected.

Received: Hello

Received: How are you?

Client disconnected.

Ex:No : 29

Creating the applications using TCP echo server and client in java/C.

TCP Chat Server:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#define PORT 8080
#define BUFFER_SIZE 1024
int main() {
   int server_fd, new_socket;
   struct sockaddr_in address;
   char buffer[BUFFER_SIZE];
   socklen_t addr_len = sizeof(address);
   // Create socket
   if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
      perror("Socket failed");
      exit(EXIT_FAILURE);
   }
   // Bind
   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);
}

// Listen
if (listen(server_fd, 1) < 0) {
    perror("Listen failed");
    close(server_fd);
    exit(EXIT_FAILURE);
}
printf("Chat server running on port %d...\n", PORT);
// Accept
if ((new_socket = accept(server_fd, (struct sockaddr *)&address, &addr_len)) < 0) {
    perror("Accept failed");
    close(server_fd);
    exit(EXIT_FAILURE);
}
printf("Client connected. Start chatting...\n");
// Chat loop
while (1) {
    // Receive message from client
    int bytes_received = read(new_socket, buffer, BUFFER_SIZE - 1);
    if (bytes_received <= 0) {
        printf("Client disconnected.\n");
        break;
    }
    buffer[bytes_received] = '\0';
    printf("Client: %s", buffer);

    // Exit condition
    if (strncmp(buffer, "exit", 4) == 0)
```

```
            break;
        // Send message to client
        printf("Server: ");
        fgets(buffer, BUFFER_SIZE, stdin);
        send(new_socket, buffer, strlen(buffer), 0);
        if (strncmp(buffer, "exit", 4) == 0)
            break;
    }
    close(new_socket);
    close(server_fd);
    return 0;
}
```

TCP Chat Client

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#define SERVER_IP "127.0.0.1"
#define PORT 8080
#define BUFFER_SIZE 1024

int main() {
    int sock;
    struct sockaddr_in server_addr;
    char buffer[BUFFER_SIZE];
    // Create socket
    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }
```

```
// 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++)
{
```