VASAVI COLLEGE OF ENGINEERING

(AUTONOMOUS)

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad – 500 031.

DEPARTMENT OF: COMPUTER SCIENCE AND ENGINEERING

NAME OF THE LABORATORY :COMPUTER NETWORKS LAB

Name:K.S.I.SivaniRoll No: 1602-21-733-052 Page No. :___

Server code:

```python
import socket

# Define the server's IP address and port
server_ip = '127.0.0.1'
server_port = 2345

# Create a socket object
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Bind the socket to the IP address and port
server_socket.bind((server_ip, server_port))

# Listen for incoming connections
server_socket.listen(5)
print(f"Server is listening on {server_ip}:{server_port}")

# Accept incoming connection
client_socket, client_address = server_socket.accept()
print(f"Accepted connection from {client_address}")

while True:
    # Receive data from the client
    data = client_socket.recv(1024).decode('utf-8')
    if not data:
        break

    print(f"Received data: {data}")

    # Send a response to the client
    response = f"Server received your message: {data}"
    client_socket.send(response.encode('utf-8'))

# Close the connection
client_socket.close()
server_socket.close()
```

VASAVI COLLEGE OF ENGINEERING

(AUTONOMOUS)

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad – 500 031.

DEPARTMENT OF: COMPUTER SCIENCE AND ENGINEERING

NAME OF THE LABORATORY :COMPUTER NETWORKS LAB

Name:K.S.I.SivaniRoll No: 1602-21-733-052 Page No. :___

CLIENT CODE:

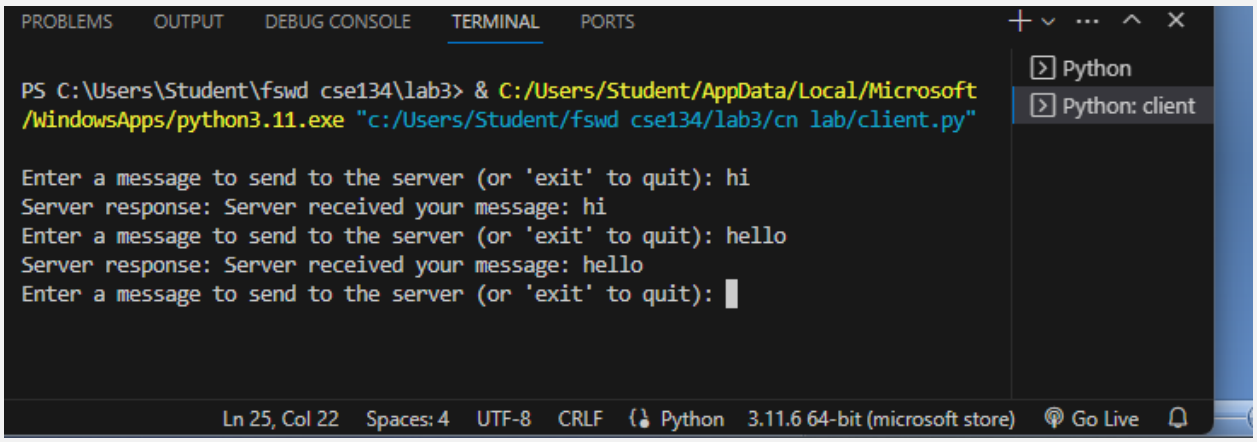
```
import socket
# Define the server's IP address and port
server_ip = '127.0.0.1'
server_port = 2345
# Create a socket object
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# Connect to the server
client_socket.connect((server_ip, server_port))

while True:
    # Send a message to the server
    message = input("Enter a message to send to the server (or 'exit'
to quit): ")
    if message.lower() == 'exit':
        break
    client_socket.send(message.encode('utf-8'))

    # Receive a response from the server
    response = client_socket.recv(1024).decode('utf-8')
    print(f"Server response: {response}")

# Close the connection
client_socket.close()
```

OUTPUT:



VASAVI COLLEGE OF ENGINEERING

(AUTONOMOUS)

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad – 500 031.

DEPARTMENT OF: COMPUTER SCIENCE AND ENGINEERING

NAME OF THE LABORATORY :COMPUTER NETWORKS LAB

Name:K.S.I.SivaniRoll No: 1602-21-733-052 Page No. :___

CRC USING SERVER AND CLIENT PROGRAMS:

SERVER.C:

```c
#include <stdio.h>
#include <string.h>
#include <stdint.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>

// CRC-32 calculation function (simplified)
uint32_t calculateCRC(const char* data, size_t len) {
    uint32_t crc = 0xFFFFFFFF;
    for (size_t i = 0; i < len; i++) {
        crc ^= data[i];
        for (int j = 0; j < 8; j++) {
            crc = (crc >> 1) ^ ((crc & 1) ? 0xEDB88320 : 0);
        }
    }
    return ~crc;
}

int main() {
    int serverSocket, clientSocket;
    struct sockaddr_in serverAddr, clientAddr;
    socklen_t addrLen = sizeof(struct sockaddr_in);

    serverSocket = socket(AF_INET, SOCK_STREAM, 0);
    if (serverSocket < 0) {
        perror("socket");
        exit(1);
    }

    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(12345);
    serverAddr.sin_addr.s_addr = INADDR_ANY;
```

```
    if (bind(serverSocket, (struct sockaddr*)&serverAddr,
sizeof(serverAddr)) < 0) {
        perror("bind");
        exit(1);
    }

    listen(serverSocket, 5);

    printf("Server is waiting for a connection...\n");
    clientSocket = accept(serverSocket, (struct sockaddr*)&clientAddr,
&addrLen);

    if (clientSocket < 0) {
        perror("accept");
        exit(1);
    }

    char buffer[1024];
    ssize_t bytesReceived = recv(clientSocket, buffer, sizeof(buffer),
0);
    if (bytesReceived < 0) {
        perror("recv");
        exit(1);
    }

    // Calculate CRC for received data
    uint32_t receivedCRC;
    ssize_t crcBytesReceived = recv(clientSocket, &receivedCRC,
sizeof(uint32_t), 0);
    if (crcBytesReceived < 0) {
        perror("recv");
        exit(1);
    }

    uint32_t calculatedCRC = calculateCRC(buffer, bytesReceived);

    // Compare the received CRC with the calculated CRC
    if (receivedCRC == calculatedCRC) {
```

```
        printf("CRC-32: 0x%08X (Matched)\n", receivedCRC);
    } else {
        printf("CRC-32: 0x%08X (Mismatch)\n", receivedCRC);
    }

    close(clientSocket);
    close(serverSocket);

    return 0;
}
```

CLIENT.C:

```
#include <stdio.h>
#include <string.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdlib.h>

// CRC-32 calculation function (simplified)
uint32_t calculateCRC(const char* data, size_t len) {
    uint32_t crc = 0xFFFFFFFF;
    for (size_t i = 0; i < len; i++) {
        crc ^= data[i];
        for (int j = 0; j < 8; j++) {
            crc = (crc >> 1) ^ ((crc & 1) ? 0xEDB88320 : 0);
        }
    }
    return ~crc;
}

int main() {
    int clientSocket;
    struct sockaddr_in serverAddr;

    clientSocket = socket(AF_INET, SOCK_STREAM, 0);
    if (clientSocket < 0) {
        perror("socket");
```

```
        exit(1);
    }

    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(12345);
    serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");

    if (connect(clientSocket, (struct sockaddr*)&serverAddr,
sizeof(serverAddr)) < 0) {
        perror("connect");
        exit(1);
    }

    // Define the data to be sent
    const char* data = "Hello, CRC!";

    // Calculate CRC for the data
    uint32_t crc = calculateCRC(data, strlen(data));

    // Send the data to the server
    send(clientSocket, data, strlen(data), 0);

    // Send the CRC value to the server
    send(clientSocket, &crc, sizeof(uint32_t), 0);

    close(clientSocket);

    return 0;
}
```

OUTPUT: