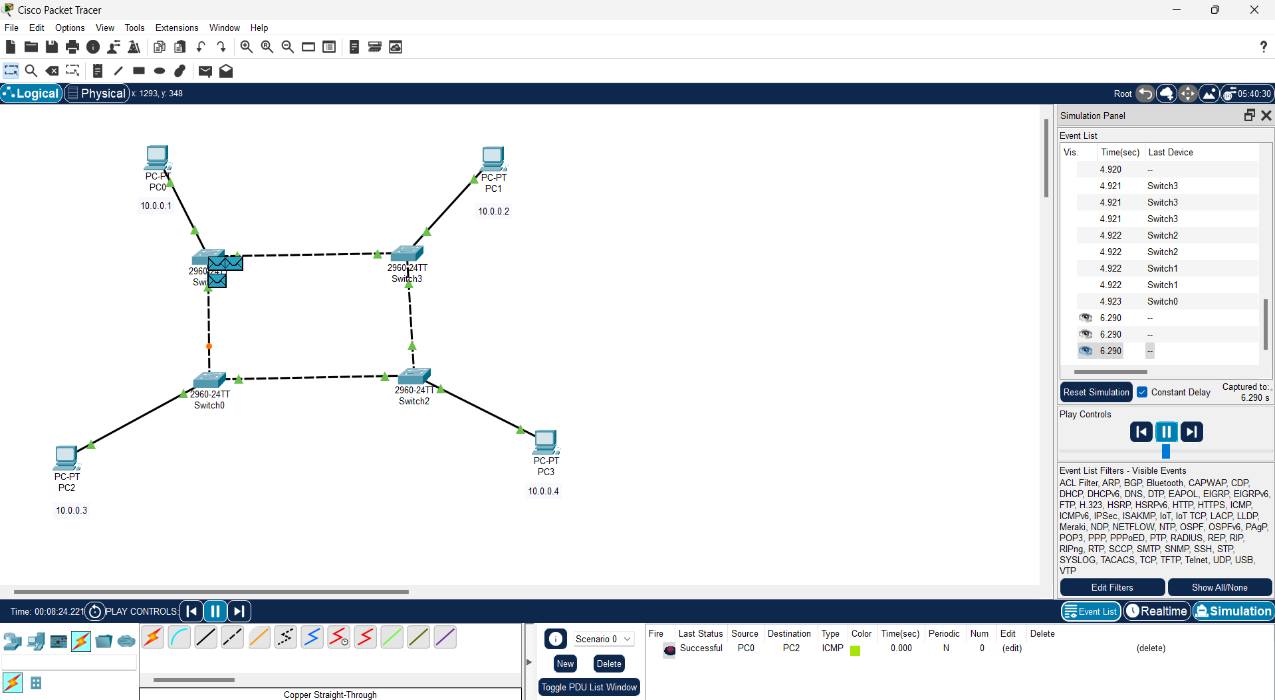
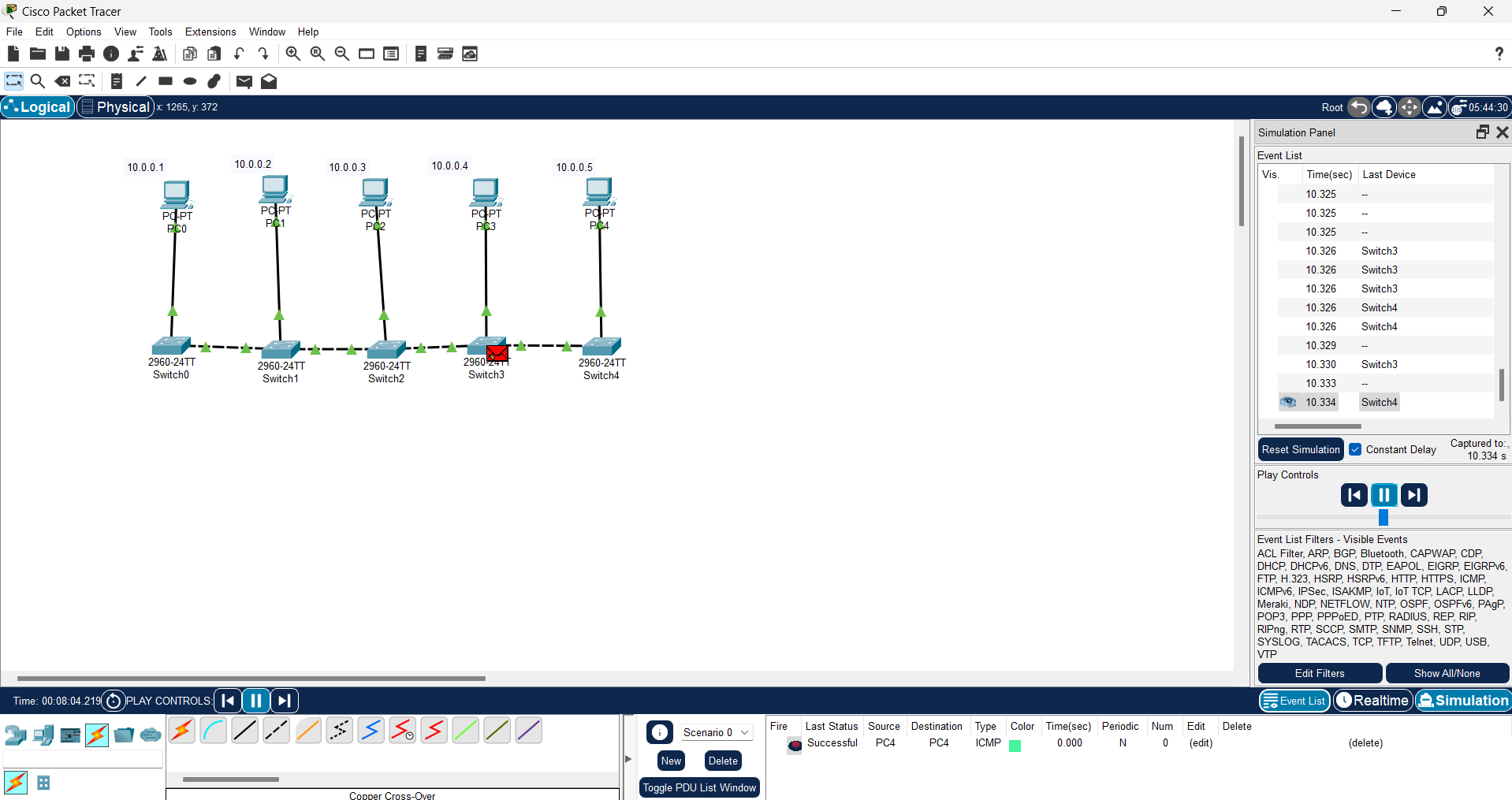
**EXP-1**

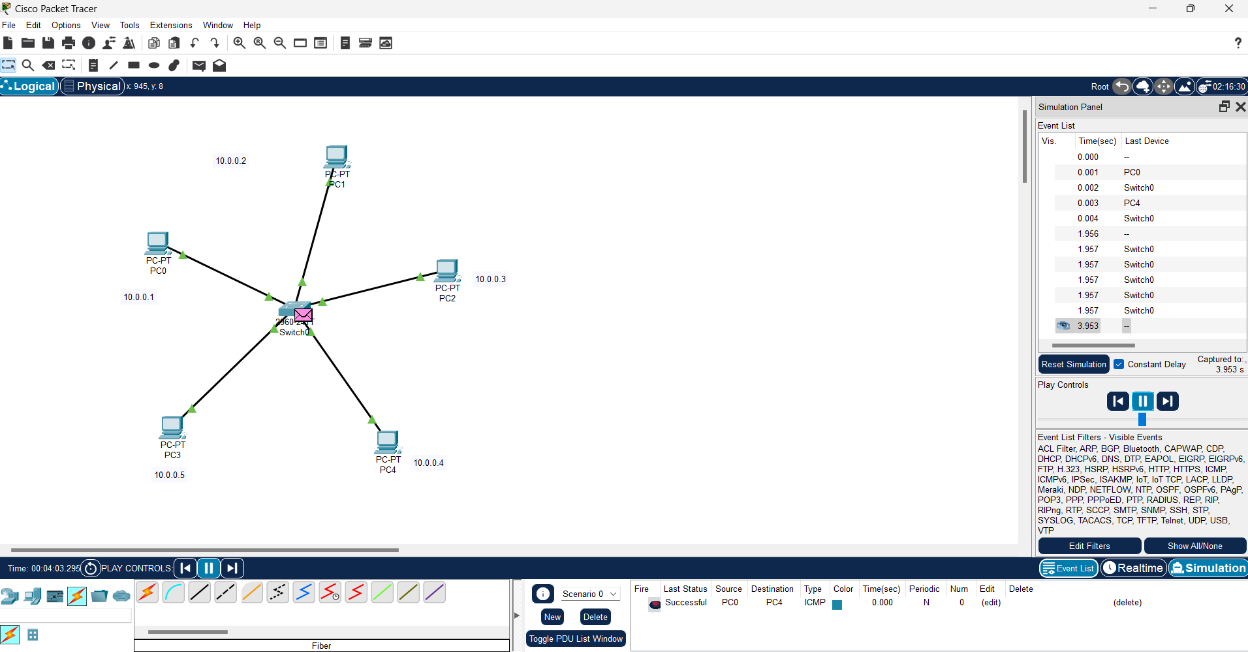
**RING TOPOLOGY**

****

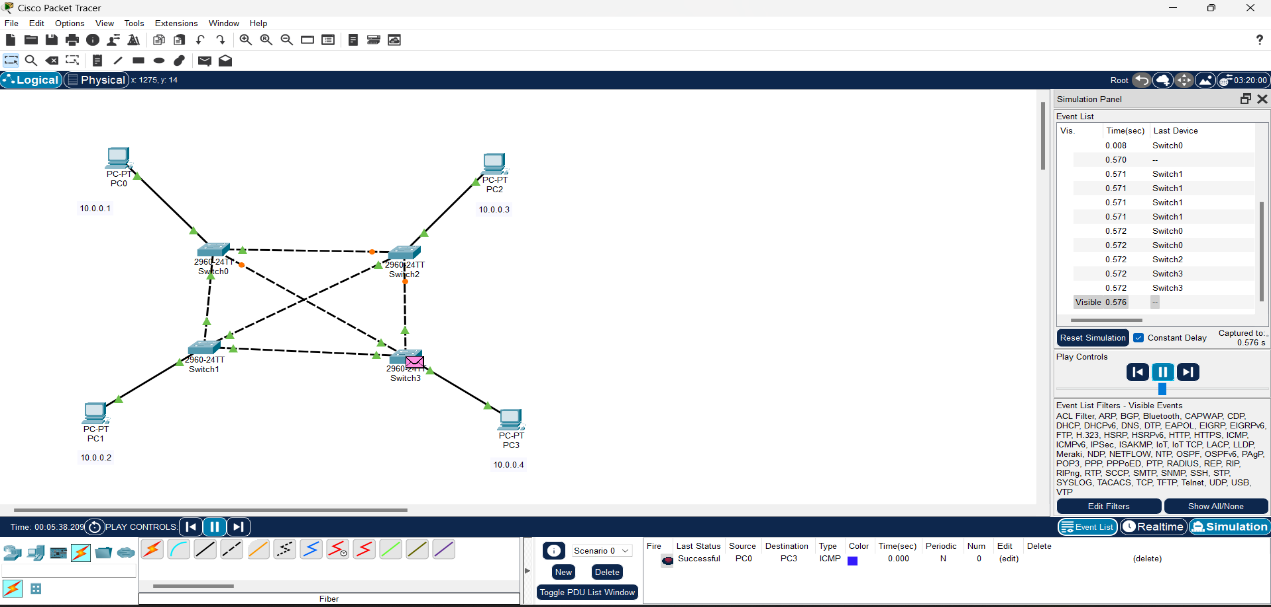
**BUS TOPOLOGY**



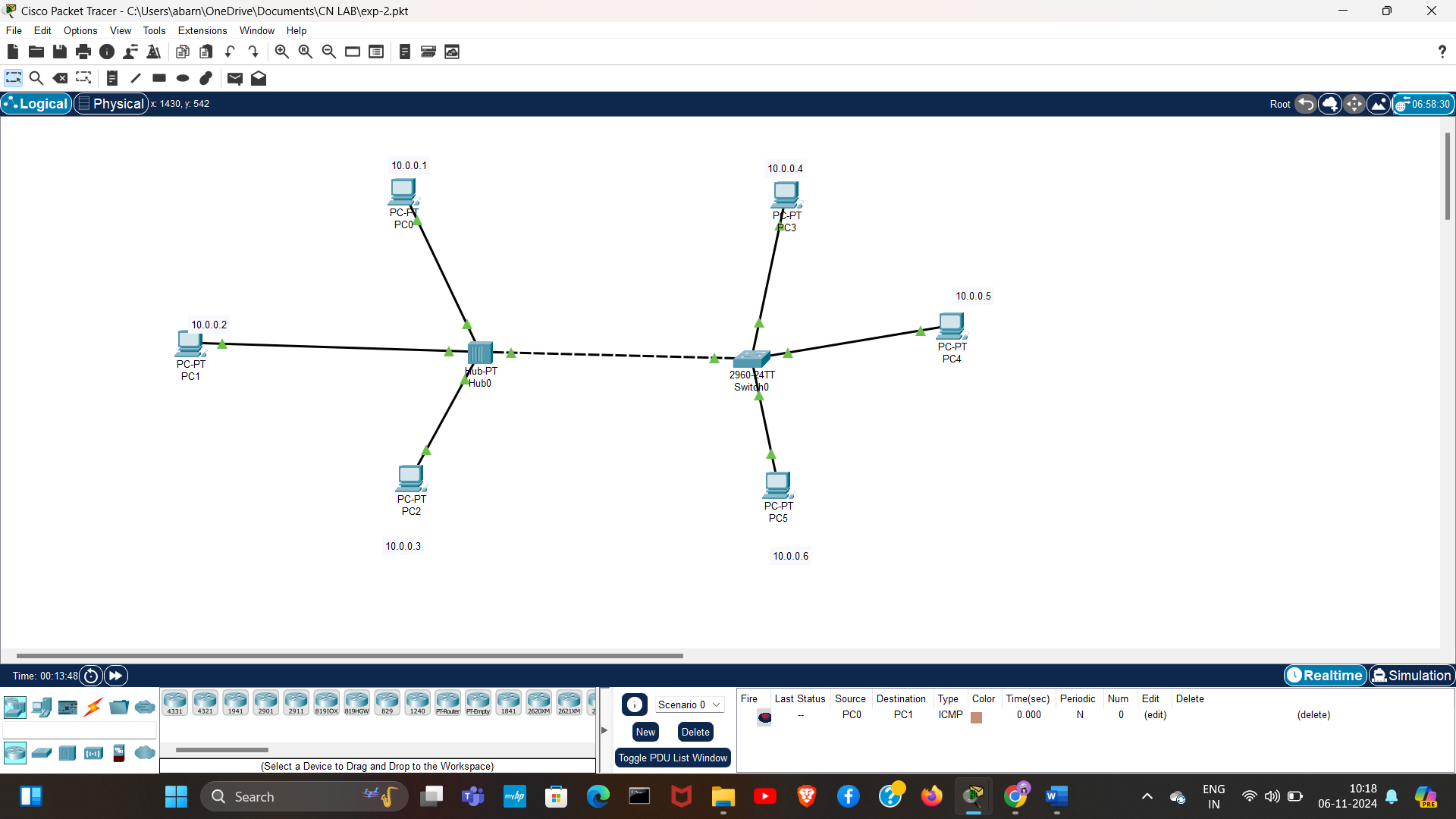
**STAR TOPOLOGY**



**MESH TOPOLOGY**



EXP-2



EXP-3

Bit:

#include <stdio.h>

#include <string.h>

void bitStuffing(char \*data, char \*stuffedData) {

int i, count = 0;

int j = 0; // Index for stuffedData

for (i = 0; data[i] != '\0'; i++) {

stuffedData[j] = data[i];

j++;

if (data[i] == '1') {

count++;

if (count == 5) {

stuffedData[j] = '0'; // Insert '0' after five consecutive '1's

j++;

count = 0;

}

} else {

count = 0;

}

}

stuffedData[j] = '\0';

}

int main() {

char data[100], stuffedData[200];

printf("Enter binary data: ");

scanf("%s", data);

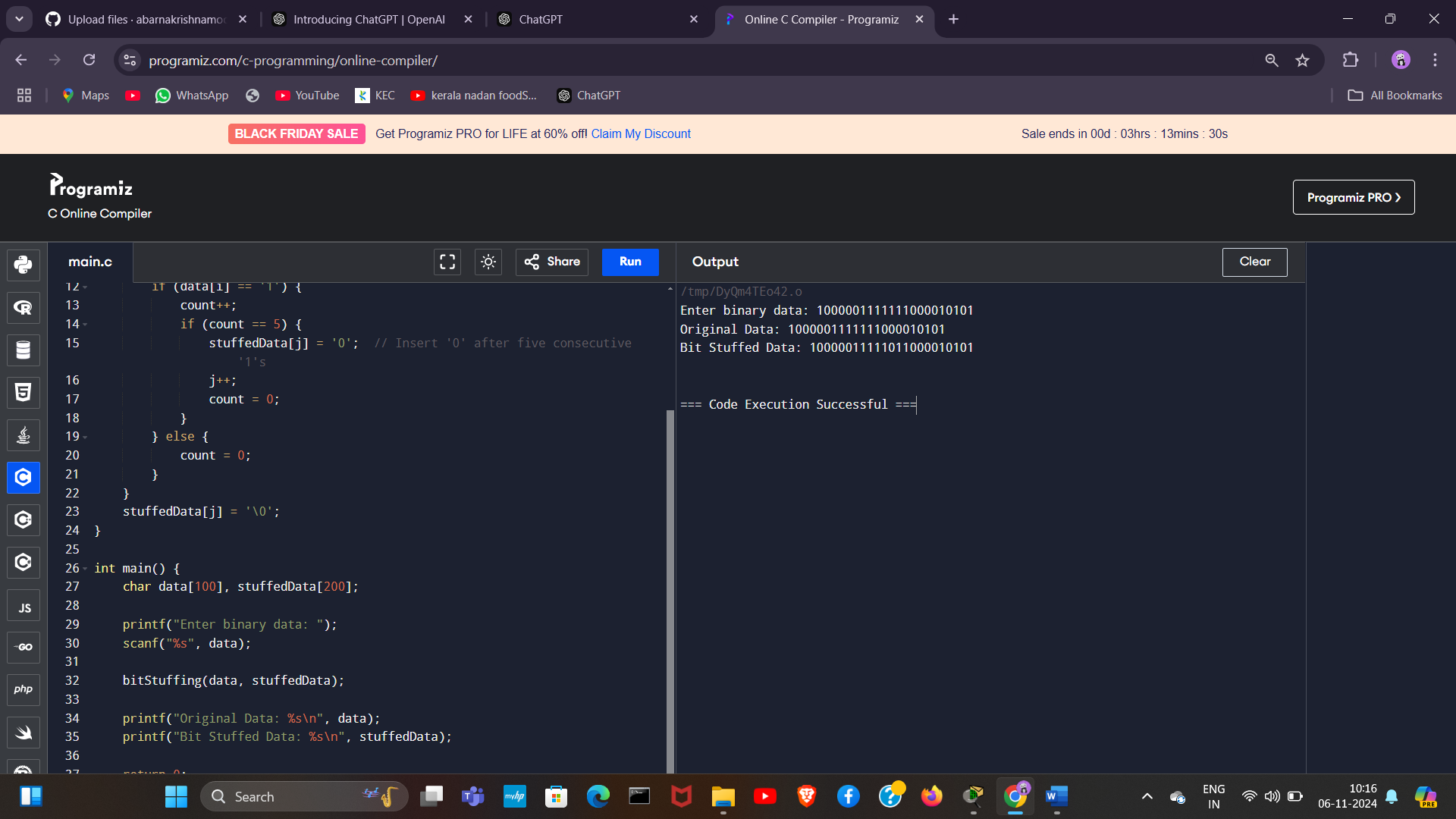
bitStuffing(data, stuffedData);

printf("Original Data: %s\n", data);

printf("Bit Stuffed Data: %s\n", stuffedData);

return 0;

}



Byte:

#include <stdio.h>

#include <string.h>

#define ESC 0x1B

#define FLAG 0x7E

void byteStuffing(char \*data, char \*stuffedData) {

int i, j = 0;

for (i = 0; data[i] != '\0'; i++) {

if (data[i] == FLAG || data[i] == ESC) {

stuffedData[j] = ESC; // Insert ESC character

j++;

}

stuffedData[j] = data[i];

j++;

}

stuffedData[j] = '\0';

}

int main() {

char data[100], stuffedData[200];

printf("Enter data (characters only, for simplicity): ");

scanf("%s", data);

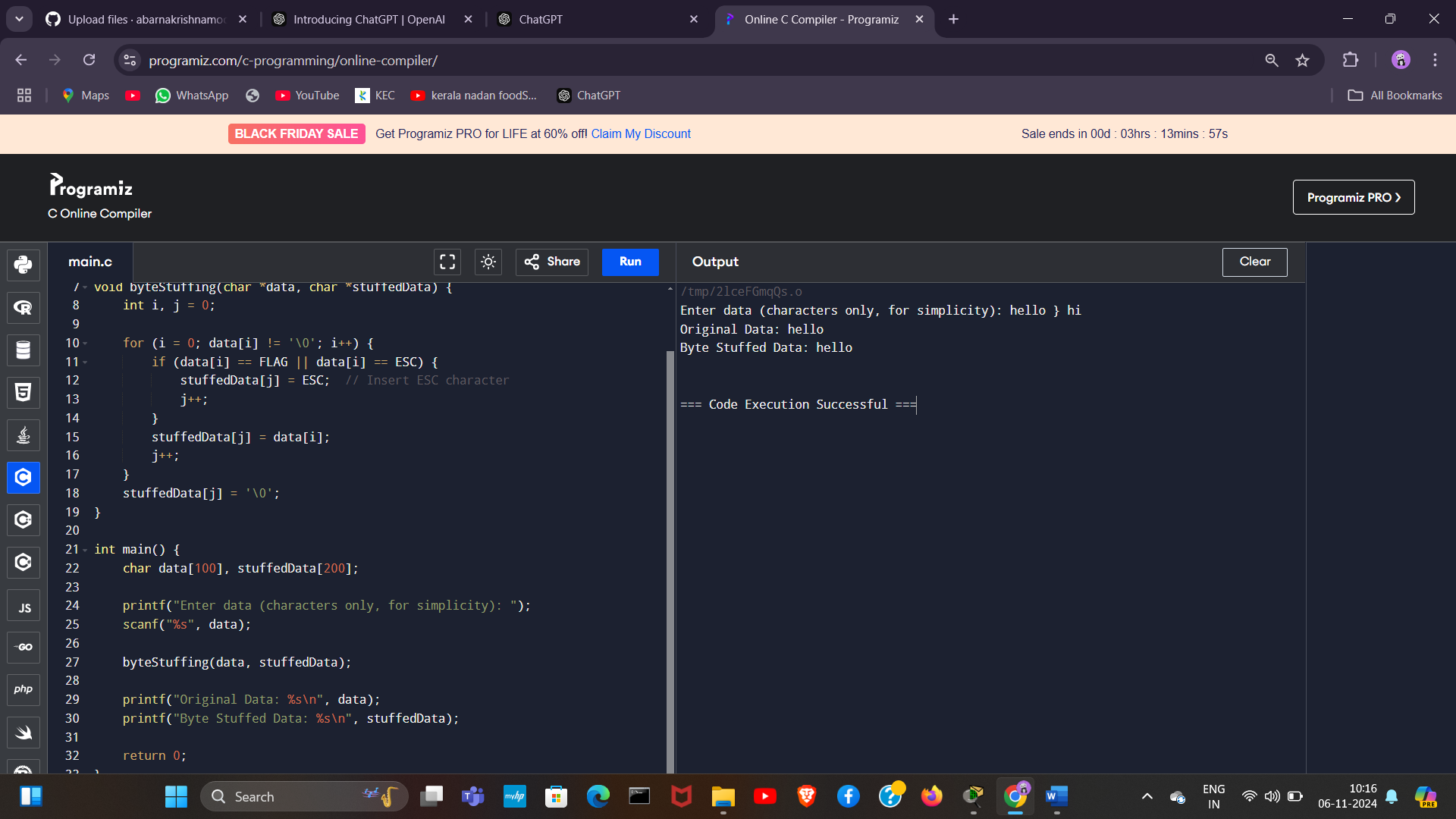
byteStuffing(data, stuffedData);

printf("Original Data: %s\n", data);

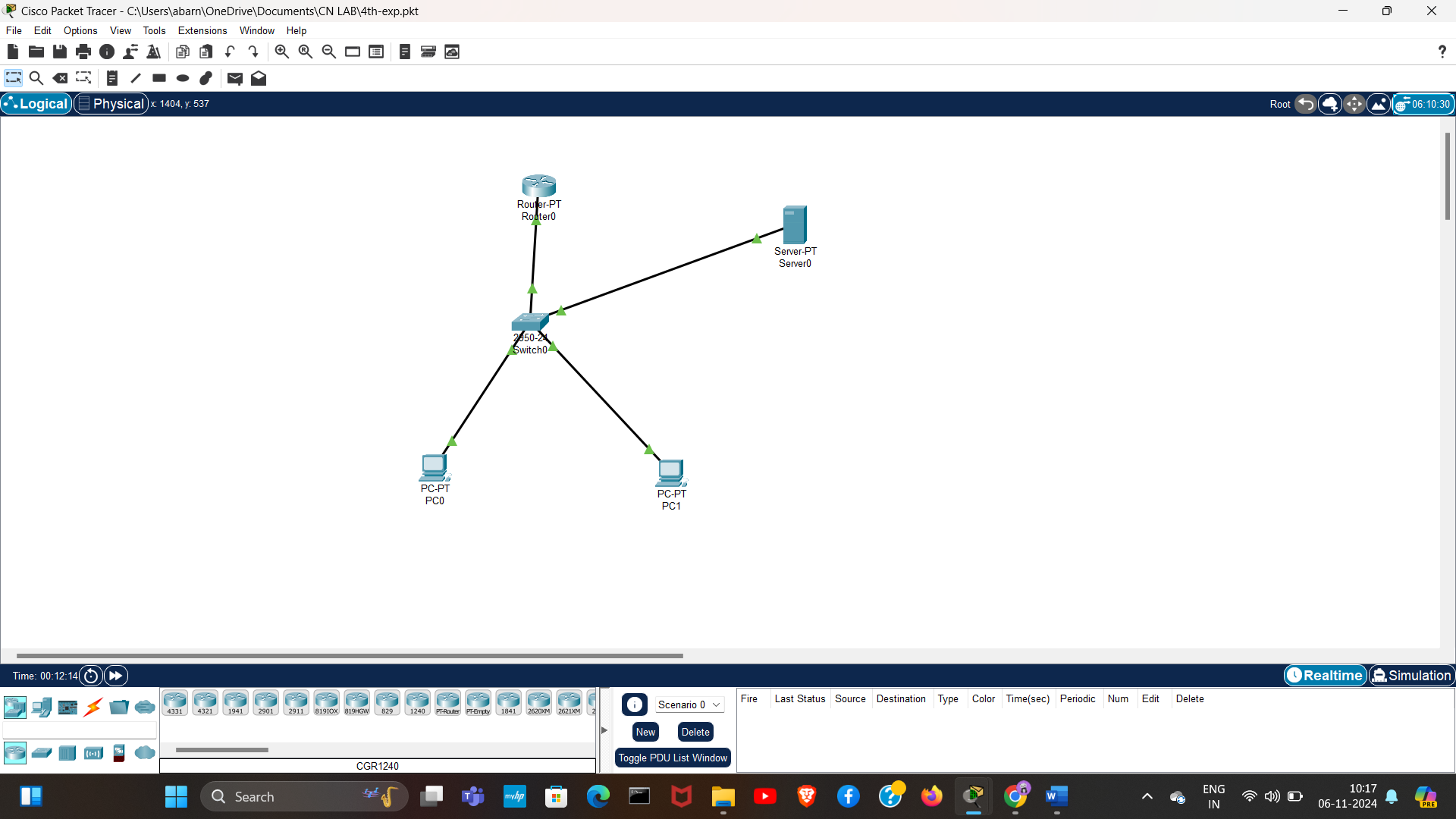
printf("Byte Stuffed Data: %s\n", stuffedData);

return 0;

}



EXP-4



EXP-6

UDP:

SERVER.C

// server\_udp\_multiclient.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <winsock2.h>

#include <ws2tcpip.h>

#pragma comment(lib, "ws2\_32.lib") // Link with Winsock library

#define PORT 8081

// Function to calculate the sum of squares

int sum\_of\_squares(int n) {

int sum = 0;

for (int i = 1; i <= n; i++) {

sum += i \* i;

}

return sum;

}

int main() {

WSADATA wsa;

SOCKET server\_fd;

struct sockaddr\_in server\_addr, client\_addr;

int addrlen = sizeof(client\_addr);

char buffer[1024] = {0};

int number, result;

// Initialize Winsock

printf("Initializing Winsock...\n");

if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0) {

printf("Failed. Error Code: %d\n", WSAGetLastError());

return 1;

}

printf("Winsock Initialized.\n");

// Create socket for UDP

if ((server\_fd = socket(AF\_INET, SOCK\_DGRAM, 0)) == INVALID\_SOCKET) {

printf("Socket creation failed. Error Code: %d\n", WSAGetLastError());

WSACleanup();

return 1;

}

// Set up the server address structure

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_addr.s\_addr = INADDR\_ANY;

server\_addr.sin\_port = htons(PORT);

// Bind the socket to the port

if (bind(server\_fd, (struct sockaddr \*)&server\_addr, sizeof(server\_addr)) == SOCKET\_ERROR) {

printf("Bind failed. Error Code: %d\n", WSAGetLastError());

closesocket(server\_fd);

WSACleanup();

return 1;

}

printf("Waiting for messages...\n");

// Main loop to handle client requests

while (1) {

memset(buffer, 0, sizeof(buffer)); // Clear buffer

int recv\_len = recvfrom(server\_fd, buffer, sizeof(buffer), 0, (struct sockaddr \*)&client\_addr, &addrlen);

// Exit loop if client sends "exit"

if (strcmp(buffer, "exit") == 0) {

printf("Client disconnected.\n");

break;

}

number = atoi(buffer); // Convert received message to number

printf("Received number from client %s:%d: %d\n", inet\_ntoa(client\_addr.sin\_addr), ntohs(client\_addr.sin\_port));

result = sum\_of\_squares(number); // Calculate sum of squares

// Send result back to client

sprintf(buffer, "%d", result);

sendto(server\_fd, buffer, strlen(buffer), 0, (struct sockaddr \*)&client\_addr, addrlen);

printf("Sent sum of squares: %d to client %s:%d\n", result, inet\_ntoa(client\_addr.sin\_addr), ntohs(client\_addr.sin\_port));

}

// Close socket and clean up

closesocket(server\_fd);

WSACleanup();

return 0;

}

CLIENT.C

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <winsock2.h>

#include <ws2tcpip.h>

#pragma comment(lib, "ws2\_32.lib") // Link with Winsock library

#define PORT 8081

int main() {

WSADATA wsa;

SOCKET sock;

struct sockaddr\_in serv\_addr;

char buffer[1024] = {0};

char input[1024];

// Initialize Winsock

printf("Initializing Winsock...\n");

if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0) {

printf("Failed. Error Code: %d\n", WSAGetLastError());

return 1;

}

printf("Winsock Initialized.\n");

// Create socket

if ((sock = socket(AF\_INET, SOCK\_DGRAM, 0)) == INVALID\_SOCKET) {

printf("Socket creation failed. Error Code: %d\n", WSAGetLastError());

WSACleanup();

return 1;

}

// Set up server address structure

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_port = htons(PORT);

// Convert IP address from text to binary form (127.0.0.1 for localhost)

if (inet\_pton(AF\_INET, "127.0.0.1", &serv\_addr.sin\_addr) <= 0) {

printf("Invalid address/ Address not supported\n");

closesocket(sock);

WSACleanup();

return 1;

}

// Main loop to send numbers to the server

while (1) {

printf("Enter a number (or type 'exit' to quit): ");

fgets(input, sizeof(input), stdin);

input[strcspn(input, "\n")] = 0; // Remove newline character

// If the user types 'exit', break the loop and close the socket

if (strcmp(input, "exit") == 0) {

sendto(sock, input, strlen(input), 0, (struct sockaddr \*)&serv\_addr, sizeof(serv\_addr));

break;

}

// Send the input number to the server

sendto(sock, input, strlen(input), 0, (struct sockaddr \*)&serv\_addr, sizeof(serv\_addr));

// Read the response from the server (sum of squares)

memset(buffer, 0, sizeof(buffer));

int addrlen = sizeof(serv\_addr);

recvfrom(sock, buffer, sizeof(buffer), 0, (struct sockaddr \*)&serv\_addr, &addrlen);

printf("Sum of squares: %s\n", buffer);

}

closesocket(sock);

WSACleanup();

return 0;

}

EXP-5

TCP:

SERVER.C

// server.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <winsock2.h>

#include <ws2tcpip.h>

#pragma comment(lib, "ws2\_32.lib") // Link with Winsock library

#define PORT 8081

// Function to calculate the sum of squares

int sum\_of\_squares(int n) {

int sum = 0;

for (int i = 1; i <= n; i++) {

sum += i \* i;

}

return sum;

}

int main() {

WSADATA wsa;

SOCKET server\_fd, new\_socket;

struct sockaddr\_in address;

int addrlen = sizeof(address);

char buffer[1024] = {0};

int number, result;

// Initialize Winsock

printf("Initializing Winsock...\n");

if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0) {

printf("Failed. Error Code: %d\n", WSAGetLastError());

return 1;

}

printf("Winsock Initialized.\n");

// Create socket

if ((server\_fd = socket(AF\_INET, SOCK\_STREAM, 0)) == INVALID\_SOCKET) {

printf("Socket creation failed. Error Code: %d\n", WSAGetLastError());

WSACleanup();

return 1;

}

// Set up the server address structure

address.sin\_family = AF\_INET;

address.sin\_addr.s\_addr = INADDR\_ANY;

address.sin\_port = htons(PORT);

// Bind the socket to the port

if (bind(server\_fd, (struct sockaddr \*)&address, sizeof(address)) == SOCKET\_ERROR) {

printf("Bind failed. Error Code: %d\n", WSAGetLastError());

closesocket(server\_fd);

WSACleanup();

return 1;

}

// Start listening for client connections

if (listen(server\_fd, 3) == SOCKET\_ERROR) {

printf("Listen failed. Error Code: %d\n", WSAGetLastError());

closesocket(server\_fd);

WSACleanup();

return 1;

}

printf("Waiting for connections...\n");

// Accept a client connection

if ((new\_socket = accept(server\_fd, (struct sockaddr \*)&address, &addrlen)) == INVALID\_SOCKET) {

printf("Accept failed. Error Code: %d\n", WSAGetLastError());

closesocket(server\_fd);

WSACleanup();

return 1;

}

// Main loop to handle client requests

while (1) {

memset(buffer, 0, sizeof(buffer)); // Clear buffer

int valread = recv(new\_socket, buffer, sizeof(buffer), 0);

// Exit loop if client sends "exit"

if (strcmp(buffer, "exit") == 0) {

printf("Client disconnected.\n");

break;

}

number = atoi(buffer); // Convert received message to number

printf("Received number: %d\n", number);

result = sum\_of\_squares(number); // Calculate sum of squares

// Send result back to client

sprintf(buffer, "%d", result);

send(new\_socket, buffer, strlen(buffer), 0);

printf("Sent sum of squares: %d\n", result);

}

// Close sockets and clean up

closesocket(new\_socket);

closesocket(server\_fd);

WSACleanup();

return 0;

}

CLIENT.C

// client.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <winsock2.h>

#include <ws2tcpip.h>

#pragma comment(lib, "ws2\_32.lib") // Link with Winsock library

#define PORT 8081

int main() {

WSADATA wsa;

SOCKET sock;

struct sockaddr\_in serv\_addr;

char buffer[1024] = {0};

char input[1024];

// Initialize Winsock

printf("Initializing Winsock...\n");

if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0) {

printf("Failed. Error Code: %d\n", WSAGetLastError());

return 1;

}

printf("Winsock Initialized.\n");

// Create socket

if ((sock = socket(AF\_INET, SOCK\_STREAM, 0)) == INVALID\_SOCKET) {

printf("Socket creation failed. Error Code: %d\n", WSAGetLastError());

WSACleanup();

return 1;

}

// Set up server address structure

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_port = htons(PORT);

// Convert IP address from text to binary form (127.0.0.1 for localhost)

if (inet\_pton(AF\_INET, "127.0.0.1", &serv\_addr.sin\_addr) <= 0) {

printf("Invalid address/ Address not supported\n");

closesocket(sock);

WSACleanup();

return 1;

}

// Connect to the server

if (connect(sock, (struct sockaddr \*)&serv\_addr, sizeof(serv\_addr)) < 0) {

printf("Connection failed. Error Code: %d\n", WSAGetLastError());

closesocket(sock);

WSACleanup();

return 1;

}

// Main loop to send numbers to the server

while (1) {

printf("Enter a number (or type 'exit' to quit): ");

fgets(input, sizeof(input), stdin);

input[strcspn(input, "\n")] = 0; // Remove newline character

// If the user types 'exit', break the loop and close the socket

if (strcmp(input, "exit") == 0) {

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

break;

}

// Send the input number to the server

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

// Read the response from the server (sum of squares)

memset(buffer, 0, sizeof(buffer));

recv(sock, buffer, sizeof(buffer), 0);

printf("Sum of squares: %s\n", buffer);

}

// Close socket and clean up

closesocket(sock);

WSACleanup();

return 0;

}