DS Assignment 3

Name: Himani Verma

Admission No: U19CS075

Implement echo client-server message passing application. Message sent from client should be displayed on server and then program should terminate.

1. Write a server (TCP) C Program that opens a listening socket and waits to serve client.

2. Write a client (TCP) C Program that connects with the server program knowing IP address and port number.

3. Get the input string from console on client and send it to server, server displays the same string.

Source Code:

Server Side:

```
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#define MAX 80
#define PORT 8080
#define SA struct sockaddr


void func(int connfd)
{
        char buff[MAX];
        int n;
        // infinite loop for chat
        for (;;) {
                bzero(buff, MAX);
```

```
		// read the message from client and copy it in buffer
		read(connfd, buff, sizeof(buff));
		// print buffer which contains the client contents
		printf("From client: %s\t To client : ", buff);
		bzero(buff, MAX);
		n = 0;
		// copy server message in the buffer
		while ((buff[n++] = getchar()) != '\n');

		// and send that buffer to client
		write(connfd, buff, sizeof(buff));

		if (strncmp("exit", buff, 4) == 0) {
			printf("Server Exit...\n");
			break;
		}
	}
}

int main()
{
	int sockfd, connfd, len;
	struct sockaddr_in servaddr, cli;
	sockfd = socket(AF_INET, SOCK_STREAM, 0);
	if (sockfd == -1) {
		printf("socket creation failed...\n");
		exit(0);
	}
	else
		printf("Socket successfully created..\n");
```

```
	bzero(&servaddr, sizeof(servaddr));

	// assign IP, PORT
	servaddr.sin_family = AF_INET;
	servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
	servaddr.sin_port = htons(PORT);

	// Binding newly created socket to given IP and verification
	if ((bind(sockfd, (SA*)&servaddr, sizeof(servaddr))) != 0) {
		printf("socket bind failed...\n");
		exit(0);
	}
	else
		printf("Socket successfully binded..\n");

	// Now server is ready to listen and verification
	if ((listen(sockfd, 5)) != 0) {
		printf("Listen failed...\n");
		exit(0);
	}
	else
		printf("Server listening..\n");
	len = sizeof(cli);
	connfd = accept(sockfd, (SA*)&cli, &len); // Accept the data packet from client and
verification
	if (connfd < 0) {
		printf("server accept failed...\n");
		exit(0);
	}
	else
```

```
        printf("server accept the client...\n");


    func(connfd); // Function for chatting between client and server
    close(sockfd); // After chatting close the socket
}
```

Client Side:

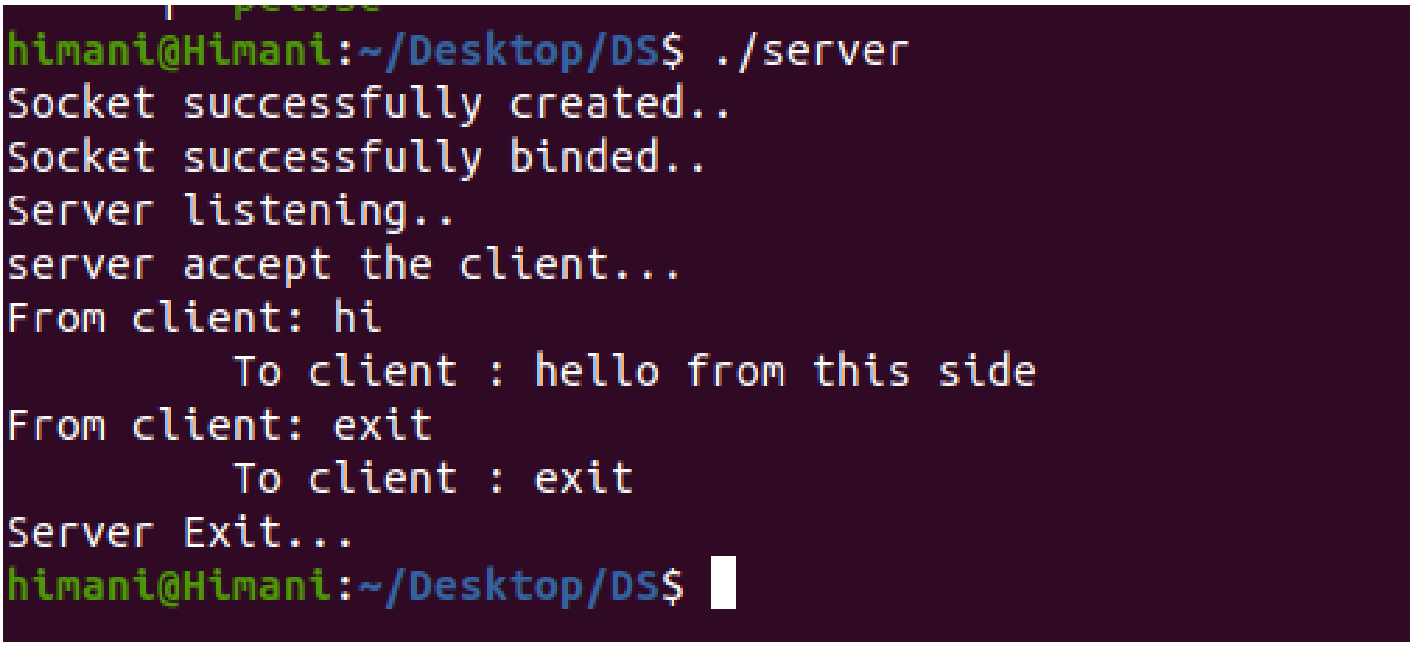
```
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#define MAX 80
#define PORT 8080
#define SA struct sockaddr
void func(int sockfd)
{
    char buff[MAX];
    int n;
    for (;;) {
        bzero(buff, sizeof(buff));
        printf("Enter the string : ");
        n = 0;
        while ((buff[n++] = getchar()) != '\n')
            ;
        write(sockfd, buff, sizeof(buff));
        bzero(buff, sizeof(buff));
        read(sockfd, buff, sizeof(buff));
        printf("From Server : %s", buff);
        if ((strncmp(buff, "exit", 4)) == 0) {
            printf("Client Exit...\n");
```

```
                        break;
                }
        }
}

int main()
{
        int sockfd, connfd;
        struct sockaddr_in servaddr, cli;
        sockfd = socket(AF_INET, SOCK_STREAM, 0);
        if (sockfd == -1) {
                printf("socket creation failed...\n");
                exit(0);
        }
        else
                printf("Socket successfully created..\n");
        bzero(&servaddr, sizeof(servaddr));

        // assign IP, PORT
        servaddr.sin_family = AF_INET;
        servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
        servaddr.sin_port = htons(PORT);

        // connect the client socket to server socket
        if (connect(sockfd, (SA*)&servaddr, sizeof(servaddr)) != 0) {
                printf("connection with the server failed...\n");
                exit(0);
        }
        else
                printf("connected to the server..\n");
```

```
        func(sockfd); // function for chat
        close(sockfd);
}
```

Output:

Server Side:



Client Side: