Assignment 1
(TASK 1)

Introduction

Most inter-process communication uses the client server model.

The client process connects to the server process typically to make a request for information.

Sockets provide the communication mechanism between two computers using TCP/UDP.

Stream sockets use TCP which is a reliable, stream-oriented protocol, and datagram sockets use UDP, which is unreliable and message oriented.

Creating Socket On Server Side-

Create a socket with the socket() system call.

Bind the socket to an address using the bind() system call. For a server socket on the Internet, an address consists of a port number on the host machine.

Listen for connections with the listen() system call.
Accept a connection with the accept() system call. This call typically blocks until a client connects with the server.

Send and receive data using read() and write() system calls.

Creating Socket On Client Side-

Create a socket with the socket() system call.
Connect the socket to the address of the server using the connect() system call.
Send and receive data using read() and write() system calls.

Problem Description

Problem 1-

Write two separate C programs, one for TCP server and one for TCP client in which server listens on some port, client connects to server sending some arbitrary message and server acknowledges it.

Problem 2-

Write two separate C program, one for TCP server (handles request for single user) and other one for client.

<u>At server side-</u>

Creates a socket and listens on some specific port to process client requests.

There is a default file in txt format having n rows where each row contains the student Roll No., student name along with their assignment marks. All the details in the rows are TAB separated. The format of the student details in the file is specified below-:

<Roll_No.>[TAB]<Student_Name>[TAB]<A_1>.[TAB]......[TAB]<A_5>

The server should be able to process the given request from the client.

1. The server should parse the string received from the client,which will contain the following requests in the given format:

    - **NUMROW:** Returns the total number of rows in the file. This is a helper command for the client to know the total no. of students info present in the file.
    - **INSERT <message>**: Inserts the string specified in message to the end of the file. The message should be in the format as specified below-:

      <Roll_No.>[TAB]<Student_Name>[TAB]<A_1>.[TAB]......[TAB]<A_5>

    - **AVERAGE <Assignment No.>:** Returns the average of all the marks for specified assignment number.
    - **GRADEX <Roll_No.>:** Returns the student details of the specified roll

number along with their grades. The grading will be carried out according to the following criteria:

Marks	Grade
>= 90	EX
< 90 && >= 80	A
< 80 && >= 70	B
< 70 && >= 60	C
< 60 && >= 50	D
< 50 && >= 40	P
< 40	F

The output should be in the format as specified-:
<STUDENT_NAME>[TAB]<GRADE>

Note: *If roll number is not specified then return all the student details along with their grades.*

- **EXIT :** TERMINATE THE CONNECTION.

At client side -

1. The client establishes a connection with the server socket on the specified port. Then, it passes NUMROW,INSERT,AVERAGE,GRADEX,EXIT requests to the server.

2. The client takes NUMROW,INSERT,AVERAGE,GRADEX,EXIT commands from the user, along with the required parameters.

3. It constructs the request and passes it to the server

4. Fetches the output message from the server and displays it to the user

Important Instructions-

1. *The server should reject requests that do not adhere to the above format by sending an error message back to the client. Please note that the user*

should type in these exact commands at the terminal; you cannot use menu-driven control, etc. ***Error handling should be done at the server.***

2. *All the processing related to the file should be done at the server.The client side is only responsible for taking the commands from the user and displaying the results sent by the server.*

Submission Instructions:

● You should have two files for server and client, named as “server.c” and “client.c” respectively.

● Save this in a folder named in the format: If your roll no. is 22CS60R05, the folder should be 22CS60R05_A4, and the compressed file should be 22CS60R05_A4.tar.gz.

● Not adhering to these instructions can incur a penalty.

Marking Scheme:

Total Marks:	**60**
Problem 1	
Establishing connection between server and client, able to exchange messages.	15
Problem 2	
NUMROW	5
INSERT	5
AVERAGE	5
GRADEX	10
Error Handling	10