COMPUTER LABORATORY - IX
PROBLEM STATEMENTS

1TS. Echo Server : Design a distributed application which consist of a server and multiple clients using Java TCP sockets. Echo server that allows multiple clients to connect to the server sends an exact text back to the respective client that it receives.

2US. Echo Server : Design a distributed application which consist of a server and multiple clients using Java UDP sockets. The server sends an exact text back to the respective client that it receives.

2TS. Chat Application : Design a distributed Chat application which consist of a server and multiple clients using Java TCP sockets. Multiple clients can simultaneously connect to the server and chat. The server shows the particular client's name with its particular message.

3TS. Power Calculation : Design a distributed application which consist of a server and client Java TCP sockets using threads. Multiple clients can simultaneously connect to the server and send messages of the format -> (a, b) where a and bare integers and server returns the value a^b (a raised to b).

4TS. Conversion : Design a distributed application in Java which consist of a server and client TCP Sockets using threads. Multiple clients can simultaneously connect to the server and send a value in Feet. The server returns the value in Metres.

5TS. Conversion : Design a distributed application in Java which consist of a server and client TCP Sockets using threads. Multiple clients can simultaneously connect to the server and send a value in Degrees. The server returns the value in Radians.

1RMI. Design a distributed application using RMI for remote computation where client submits two strings to the server and the server returns the concatenation of the given strings.

2RMI. Design a distributed application using RMI for remote computation where client submits two strings to the server and the server verifies if string 1 is a substring of string 2 and returns a message as a result accordingly.

3RMI. Design a distributed application using RMI for remote computation where client submits two strings to the server and the server verifies if string 1 is the reverse of string 2 and returns a message as a result accordingly.

4RMI. Design a distributed application using RMI for remote computation where client submits two strings to the server and the server verifies if both string 1 and 2 contain equal number of vowels and returns a result accordingly.

5RMI. Design a distributed application using RMI for remote computation where client submits 2 integers (a and b) to the server and the server returns the value a^b (a raised to b)

1MPI. Design a distributed application using Message Passing Interface (MPI) for remote computation where client submits an integer value to the server and server returns to the client whether the integer sent is a prime number or not.

2MPI. Design a distributed application using Message Passing Interface (MPI) for remote computation where client submits an integer value to the server and the server calculates factorial and returns the result to the client program.

3MPI. Design a distributed application using Message Passing Interface (MPI) for remote computation where client submits an integer value to the server and the server calculates the reciprocal and returns the result to the client program.

4MPI. Design a distributed application using Message Passing Interface (MPI) for remote computation where client submits a value in Feet to the server and the server converts this input into Meters and sends the value to the client program.

5MPI. Design a distributed application using Message Passing Interface (MPI) for remote computation where client submits a value in Degrees to the server and the server converts this input into Radians and sends the value to the client program.

1CORBA. File Transfer Application

This application allows a client to transfer (or download) any type of file (plain text or binary) from a remote machine. The first step is to define a remote interface that specifies the signatures of the methods to be provided by the server and invoked by clients.

FileClient.java is the client program that looks up the server.

FileInterface.idl defines an interface in the OMG Interface Definition Language (IDL) for a file transfer operation.

FileServant.java implements the downloadFile method declared in FileInterface.idl.

FileServer.java implements a CORBA server that initializes the ORB, creates a FileServant object, registers the object in the CORBA Naming Service (COS Naming), prints a status message, and waits for incoming client requests.

2. CORBA. Stock price update application: Client programs often react to changes or updates that occur in a server. For example, a client graph might need to be updated with each stock price update on a stock market server. The client has two options in this scenario: (1) periodically ask for the stock price via a method request on the stock server or (2) ask to be notified by the server whenever a price change occurs. Write an application using any of the two options to update stock price using Java IDL.

3. CORBA. Accessing URL: This application finds and reads a given URL which is contained in a file named `URLReader.java`. The `URLReader.java` file contains the text for reading a given URL in Java which `reads URL and reports # of bytes reads`. IDL then accesses this data through the `URLRead` routine which is in a file. The file contains the text for reading a given URL in Java. Usage: `java URLReader <URL>`.

4.CORBA. Bank Agent: The application has `Bank.idl` file which contains an `Account` interface; provides a single method `balance()` for obtaining the current balance. The `AccountManager` interface creates an account for the user if one does not already exist. The `Client` class implements the client application which obtains the current balance of a bank account. `Server.java` file implements the Server class for the server side of the banking application.

5.CORBA. TimeServer: Build a TimeServer program as a distributed application with application clients. The TimeServer program has a single operation which returns the current time of a server machine to any client that requests it. The `Tracker.idl` has an interface called `Time` and `getTime()` operation. The client prints the time on the server.

6. CORBA. **InterestRates Server**: Build the InterestRates program as a Distributed application. From one centralized server machine, it returns the current interest rates of different types of accounts a particular bank provides. Clients can use these rates to perform their own computations. Interest rates provided in the IDL defined as follows:

```
    module Interest {
          interface Rates {
          float getHomeLoan();
          float getPersonalLoan();
          float getCarLoan();
          float getGoldLoan();     };         };
```

7. CORBA. Bank Application: Build a sample client application that can query and perform operations on the balance in a bank checking account.

The `Bank.idl` file defines the Bank module that contains two interfaces: The `Account` interface provides methods for setting and obtaining the current balance; the `CheckingsAccount` interface provides a method that opens, credits to, debits from, and creates accounts with a specified name, and returns an Account object reference. The Account object reference can then be used to obtain the balance in the account.

```
module Bank {
 interface Account {
  float getBalance ();
  void setBalance (in float balance)
         raises (InvalidTransaction);
 };
 interface CheckingsAccount {
  Account open(in string name);
  Account create(in string name, in float balance);
  Account credit(in string name, in float amount)
                      raises (InvalidTransaction);
  Account debit(in string name, in float amount)
                     raises (InvalidTransaction);
 };};
```

1. Election - Bully: Implement a Bully Election Algorithm for distributed systems in Java. With seven processes (p1 to p7) communicating where highest numbered process will be coordinator when it is down. Any of the lower number process can initiate election like p4 but the election is won by p6 if it is up else p5 if p6 down, and p5 up or p4 is winner when p5 , p6 are down.

2. Election - Ring : Implement a Ring Election Algorithm for distributed systems in Java. In this algorithm assume that the link between the process are unidirectional and every process can message to the process on its right only. Each process is associated with name, priority and state (Active/Inactive). Accept the number of processes from the user with their priorities and states. Select the process with highest priority as coordinator. Allow user to choose any active process to initialize Election.

1. Webservice-SOAP based: Implement a SOAP based web service which will accept a simple piece of text (Personname) and return another piece of text as a result to a consumer.(Hello there Personname).

2. Webservice-SOAP based: Implement a SOAP based Arithmetic web service which will accept a two values and operation to be performed (a, b, op) and return operation's result to a web based service requestor. (The result of "operation" is ....).

3. Webservice-RESTful: Implement a web service which will accept two strings of text alongwith "string manipulation functions" and return it as a result to a web based service consumer.

4. Webservice-SOAP based: Implement a SOAP based web service which will accept a temperature in Farenheight and return temperature in Celcius as a result to a consumer.

1. Publish-Subscribe System: Implement a topic based Public Subscribe System for Stock quotes.
Consider the example of stock quotes disseminated to a large number of interested brokers. In a first step, we are interested in buying stocks, advertised by stock quote events. Such events consist of five attributes: a global identifier, the name of the company, the price, the amount of stocks, and the identifier of the selling trader.

1. Microservice: Implement a ToDoManager application which will have the following two services namely the User service and the To-Do service:

User Service: The user service list the users in an application and also allows to query the user lists based on their usernames.

To-Do Service: The ToDo service lists the projects filtered on the basis of usernames.

2. Microservice: Implement a microservice that keeps track of the number of wins, losses and ties in an Online game of your choice.