1	1.1) Addition Server: Implement multi-threaded client/server Process communication using RMI for Addition of Numbers on 2 different machines.

2	1.2) Multiplication Server: Implement multi-threaded client/server Process communication using RMI for Multiplication of Numbers on 2 different machines.

3	1.3) Division Server: Implement multi-threaded client/server Process communication using RMI for Division of Numbers on 2 different machines.

4	1.4) Subtract Server: Implement multi-threaded client/server Process communication using RMI for Subtract of Numbers on 2 different machines.

5	1.5) Power Calculation: Implement multi-threaded client/server Process communication using RMI for 2's Power of Given Number on 2 different machines.

6	1.6) Celsius to Fahrenheit Conversion Server: Implement multi-threaded client/server Process communication using RMI for Celsius to Fahrenheit Conversion Server on 2 different machines.

7	1.7) Miles to Kilometer Conversion Server: Implement multi-threaded client/server Process communication using RMI for Miles to Kilometer Conversion Server on 2 different machines.

8	1.8) Echo Server: Implement multi-threaded client/server Process communication using RMI for Printing Name Appending to Hello on 2 different machines.

9	1.9) Compare 2 Strings: Implement multi-threaded client/server Process communication using RMI for Comparing 2 Strings and Return lexicographically Largest string. Perform operations on 2 different machines.

10	1.10) Count Vowels: Implement multi-threaded client/server Process communication using RMI for Counting Vowels Present in Word. Perform operations on 2 different machines.

11	1.11) Factorial of Number: Implement multi-threaded client/server Process communication using RMI for Find factorial of Number. Perform operations on 2 different machines.

12	2.1 Design a distributed application using MPI for computation where root process has an array of elements equal to the size of processors and distributed to the worker processes which calculates and displays the intermediate sums calculated at different processors. Perform these operations on 2 different machines.

13	2.2 Design a distributed application using MPI for computation where root process has an array of elements equal to the size of processors which is divided to the worker processes which calculates and displays the intermediate multiplication calculated at different processors.

14	2.3 Design a distributed application that generate a random array of numbers on the root process (process 0) using Java Message Passing Interface (MPI), Scatter the numbers to all processes, giving each process an equal amount of numbers. Each process computes the average of their subset of the numbers. Gather all averages to the root process. The root process then computes the average of these numbers to get the final average.

15	2.4 Design a distributed application using MPI for computation where root process has an array of elements equal to the size of processors which is divided to the worker processes which calculates the reciprocal and resultant array will be displayed at root.

16	3.1 Develop String Reversing distributed application using CORBA to demonstrate object brokering. Perform operations on 2 different machines.

17	3.2 Develop Changing Case of String to Uppercasing distributed application using CORBA to demonstrate object brokering. Perform operations on 2 different machines.

18	4.1 Implement Berkeley algorithm for clock synchronization using two physical machines. Perform operations on 2 different machines.

19	4.2 Implement clock synchronization with Berkeley algorithm with time daemon server using two physical machines.

20	4.3 Implement clock synchronization with Berkeley algorithm with time daemon server using two physical machines.

21	5.1 Implement token ring based mutual exclusion algorithm using 2 physical machines. Ensure that a unique token is shared among the sites. A site is allowed to enter its Critical Section if it possesses the token. Mutual exclusion is ensured because the token is unique.

22	5.2 Implement Mutual Exclusion algorithm based on Token Ring using 2 physical machines. Ensure that a unique token is shared among the sites. A site is allowed to enter its Critical Section if it possesses the token. Mutual exclusion is ensured because the token is unique.

23	5.3 Implement token ring based mutual exclusion algorithm using 2 physical machines. Ensure that a unique token is shared among the sites. A site is allowed to enter its Critical Section if it possesses the token. Mutual exclusion is ensured because the token is unique.

24	6.1 Perform Leader Election using Bully Algorithm. Also print the messages exchanged between processes.

25	6.2 Perform Leader Election using Bully Algorithm. Also print the messages exchanged between processes.

26	6.3 Perform Leader Election using Bully algorithm. Demonstrate the time complexity of an algorithm.

27	6.4 Perform Leader Election using Toke Ring Algorithm. Also print the messages exchanged between processes.

28	6.5 Perform Leader Election using Toke Ring Algorithm. Also print the messages exchanged between processes.

29	6.6 Perform Leader Election using Toke Ring Algorithm. Also print the messages exchanged between processes.

30	7.1 Create a Simple Calculator web service and consume that web service.

31	7.2 Create a Simple Interest Calculator web service and consume that web service.

32	7.3 Create a web service which takes User's Name as Input and Display Hello User_Name Entered by User and Consume that web service.

33	7.4 Create a web service which takes Miles as Input and Display Kilometer as Output and Consume that web service.

34	7.5 Create a web service which takes Miles as Input and Display Kilometer as Output and Consume that web service.