11. Write a client server program using UDP in which client sends a string to the server and server replies the reverse of the string.

- **Client side**

```
import java.net.*;
import java.util.*;
class servercli{
	public static void main(String args[]){
		try{
			DatagramSocket clientsocket = new DatagramSocket();
			InetAddress serverAddress =
InetAddress.getByName("localhost");
			int serverPort = 12345;

			Scanner sc = new Scanner(System.in);
			System.out.println("enter a string to send to the server:");
			String msg = sc.nextLine();

			byte[] sendBuffer = msg.getBytes();
			DatagramPacket sendPacket = new
DatagramPacket(sendBuffer,sendBuffer.length,serverAddress,serverPort);
			clientsocket.send(sendPacket);

			byte[] receiveBuffer = new byte[1024];
			DatagramPacket receivePacket = new
DatagramPacket(receiveBuffer,receiveBuffer.length);
			clientsocket.receive(receivePacket);

			String reversedString = new
String(receivePacket.getData(),0,receivePacket.getLength());
			System.out.println("reversed String from server: " +
reversedString);

			clientsocket.close();
		}
		catch(Exception e){
			e.printStackTrace();
		}
	}
}
```

- **Server side**

```
import java.net.*;

class rev{
	public static void main(String args[]){
		try{
```

```
                DatagramSocket sersoc = new DatagramSocket(12345);
                System.out.println("server is listening on port 12345");

                byte[] receiveBuffer = new byte[1024];
                byte[] sendBuffer;

                while(true){
                        DatagramPacket receivePacket = new
DatagramPacket(receiveBuffer,receiveBuffer.length);
                        sersoc.receive(receivePacket);

                        String receiveString = new
String(receivePacket.getData(),0,receivePacket.getLength());
                        System.out.println("received from  client: " +
receiveString);

                        String reverseString = new
StringBuilder(receiveString).reverse().toString();

                        sendBuffer = reverseString.getBytes();
                        InetAddress clientAddress =
receivePacket.getAddress();
                        int clientPort = receivePacket.getPort();

                        DatagramPacket sendPacket = new
DatagramPacket(sendBuffer,sendBuffer.length,clientAddress,clientPort);
                        sersoc.send(sendPacket);
                        System.out.println("sent to client: " + reverseString);
                }
        }
        catch(Exception e){
                e.printStackTrace();
        }
    }
}
```

12. Write a client server program using UDP in which client sends an integer number to the server and server replies the factorial of it.

- **Client side**

```
import java.net.*;
import java.util.*;

class ClientFacto {
   public static void main(String args[]) {
      try {
         DatagramSocket clientSocket = new DatagramSocket();
         InetAddress serverAddress = InetAddress.getByName("localhost");
```

```
            int serverPort = 12345;

            Scanner scanner = new Scanner(System.in);
            System.out.print("Enter an integer to send to the server: ");
            String message = scanner.nextLine();

            byte[] sendBuffer = message.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendBuffer,
sendBuffer.length, serverAddress, serverPort);
            clientSocket.send(sendPacket);

            byte[] receiveBuffer = new byte[1024];
            DatagramPacket receivePacket = new DatagramPacket(receiveBuffer,
receiveBuffer.length);
            clientSocket.receive(receivePacket);

            String response = new String(receivePacket.getData(), 0,
receivePacket.getLength());
            System.out.println("Response from server: " + response);

            clientSocket.close();
        }
                catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

- **Server side**

```
import java.net.*;
import java.util.*;

class ServerFacto {
    public static void main(String args[]) {
        try {
            DatagramSocket serverSocket = new DatagramSocket(12345);
            System.out.println("Server is listening on port 12345");

            byte[] receiveBuffer = new byte[1024];
            byte[] sendBuffer;

            while (true) {
                DatagramPacket receivePacket = new
DatagramPacket(receiveBuffer, receiveBuffer.length);
                serverSocket.receive(receivePacket);

                String receivedString = new String(receivePacket.getData(), 0,
receivePacket.getLength());
```

```
            System.out.println("Received from client: " + receivedString);

            try {

                int number = Integer.parseInt(receivedString);
                int factorial = calculateFactorial(number);

                String response = String.valueOf(factorial);
                sendBuffer = response.getBytes();
            }
                            catch (NumberFormatException e) {
                String errorResponse = "Invalid input. Please send a valid
integer.";
                sendBuffer = errorResponse.getBytes();
            }

            InetAddress clientAddress = receivePacket.getAddress();
            int clientPort = receivePacket.getPort();

            DatagramPacket sendPacket = new DatagramPacket(sendBuffer,
sendBuffer.length, clientAddress, clientPort);
            serverSocket.send(sendPacket);
            System.out.println("Response sent to client.");
        }
    }
              catch (Exception e) {
        e.printStackTrace();
    }
  }

  private static int calculateFactorial(int number) {
    int result = 1;
    for (int i = 1; i <= number; i++) {
      result *= i;
    }
    return result;
  }
}
```

13. Write a client server program using UDP in which client sends an integer number to theserver and server replies the Fibonacci series till that number.

- **Client side**

```
import java.net.*;
import java.util.*;

class clientfibo {
```

```
    public static void main(String args[]) {
        try {
            DatagramSocket clientSocket = new DatagramSocket();

            InetAddress serverAddress = InetAddress.getByName("localhost");
            int serverPort = 12345;

            Scanner sc = new Scanner(System.in);
            System.out.print("Enter the number n to get the first n Fibonacci
numbers: ");
            String input = sc.nextLine();
            int n = Integer.parseInt(input);

            byte[] sendBuffer = input.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendBuffer,
sendBuffer.length, serverAddress, serverPort);
            clientSocket.send(sendPacket);

            byte[] receiveBuffer = new byte[1024];
            DatagramPacket receivePacket = new DatagramPacket(receiveBuffer,
receiveBuffer.length);
            clientSocket.receive(receivePacket);

            String response = new String(receivePacket.getData(), 0,
receivePacket.getLength());
            System.out.println("The first " + n + " Fibonacci numbers are: " +
response);

            clientSocket.close();
        }
                catch (Exception e)
                {
            e.printStackTrace();
        }
    }
}
```

- **Server side**

```
import java.net.*;

class serverfibo {
    public static void main(String args[]) {
        try {
            DatagramSocket serverSocket = new DatagramSocket(12345);

            System.out.println("Server is waiting for client messages...");

            while (true) {
```

```
            byte[] receiveBuffer = new byte[1024];
            DatagramPacket receivePacket = new
DatagramPacket(receiveBuffer, receiveBuffer.length);

            serverSocket.receive(receivePacket);

            String message = new String(receivePacket.getData(), 0,
receivePacket.getLength());
            int n = Integer.parseInt(message);

            String fibonacciSequence = generateFibonacciSequence(n);

            InetAddress clientAddress = receivePacket.getAddress();
            int clientPort = receivePacket.getPort();
            byte[] sendBuffer = fibonacciSequence.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendBuffer,
sendBuffer.length, clientAddress, clientPort);
            serverSocket.send(sendPacket);
        }
    }
              catch (Exception e)
              {
        e.printStackTrace();
    }
  }

  public static String generateFibonacciSequence(int n) {
    StringBuilder fibonacciSequence = new StringBuilder();

    if (n <= 0) {
      return "";
    }

    int a = 0, b = 1;

    fibonacciSequence.append(a);

    for (int i = 1; i < n; i++) {
      fibonacciSequence.append(", ").append(b);
      int next = a + b;
      a = b;
      b = next;
    }
    return fibonacciSequence.toString();
  }
}
```

14. Write a client server program using UDP in which client sends two integer numbers ‘x’ and‘n’ to the server and server replies x raise to n.

- **Client side**

```
import java.net.*;
import java.io.*;

class raiseclient {
   public static void main(String args[]) {
      DatagramSocket socket = null;

      try {
         socket = new DatagramSocket();
         InetAddress serverAddress = InetAddress.getByName("localhost");
         int serverPort = 9876;

         BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));
         System.out.print("Enter the base (x): ");

         int x = Integer.parseInt(reader.readLine());
         System.out.print("Enter the exponent (n): ");

         int n = Integer.parseInt(reader.readLine());
         String message = x + ", " + n;

         byte[] sendData = message.getBytes();
         DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, serverAddress, serverPort);
         socket.send(sendPacket);

         byte[] receiveData = new byte[1024];
         DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
         socket.receive(receivePacket);

         String resultMessage = new String(receivePacket.getData(), 0,
receivePacket.getLength());
         System.out.println("Server response: " + resultMessage);
      }
            catch (Exception e)
            {
         e.printStackTrace();
      }
            finally
            {
         if (socket != null && !socket.isClosed()) {
            socket.close();
         }
      }
```

```
    }
}
```

- **Server side**

```
import java.net.*;

class raiseserver{
    public static void main(String args[]) {
        DatagramSocket socket = null;

        try {
            socket = new DatagramSocket(9876);
            System.out.println("Server is ready to receive data...");

            while (true) {
                byte[] receiveData = new byte[1024];
                DatagramPacket receivePacket = new
DatagramPacket(receiveData, receiveData.length);
                socket.receive(receivePacket);

                String message = new String(receivePacket.getData(), 0,
receivePacket.getLength());
                String[] input = message.split(",");

                int x = Integer.parseInt(input[0].trim());
                int n = Integer.parseInt(input[1].trim());
                double result = Math.pow(x, n);
                String resultMessage = "Result: " + x + " raised to " + n + " is " +
result;

                byte[] sendData = resultMessage.getBytes();
                InetAddress clientAddress = receivePacket.getAddress();
                int clientPort = receivePacket.getPort();

                DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, clientAddress, clientPort);
                socket.send(sendPacket);
            }
        }
                catch (Exception e)
                {
            e.printStackTrace();
        }
                finally
                {
            if (socket != null && !socket.isClosed()) {
                socket.close();
            }
```

```
        }
    }
}
```

15. Write a client server program using UDP in which client requests date from the server andserver sends the date.

- **Client side**

```
import java.net.*;
import java.io.*;

 class dateclient {
    public static void main(String args[]) {
        DatagramSocket socket = null;

        try {
            socket = new DatagramSocket();
            InetAddress serverAddress = InetAddress.getByName("localhost");
            int serverPort = 9876;
            String requestMessage = "GET_DATE";

            byte[] sendData = requestMessage.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, serverAddress, serverPort);
            socket.send(sendPacket);

            byte[] receiveData = new byte[1024];
            DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
            socket.receive(receivePacket);

            String currentDateTime = new String(receivePacket.getData(), 0,
receivePacket.getLength());
            System.out.println("Server response: " + currentDateTime);
        }
                catch (Exception e)
                {
            e.printStackTrace();
        }
                finally
                {
            if (socket != null && !socket.isClosed()) {
                socket.close();
            }
        }
    }
}
```

- **Server side**

```
import java.net.*;
import java.text.SimpleDateFormat;
import java.util.Date;

class dateserver {
   public static void main(String args[]) {
      DatagramSocket socket = null;

      try {
         socket = new DatagramSocket(9876);
         System.out.println("Server is ready to receive data...");

         while (true) {
            byte[] receiveData = new byte[1024];

            DatagramPacket receivePacket = new
DatagramPacket(receiveData, receiveData.length);
            socket.receive(receivePacket);
            String currentDateTime = getCurrentDateTime();

            byte[] sendData = currentDateTime.getBytes();
            InetAddress clientAddress = receivePacket.getAddress();
            int clientPort = receivePacket.getPort();

            DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, clientAddress, clientPort);
            socket.send(sendPacket);
         }
      }
               catch (Exception e)
               {
         e.printStackTrace();
      }
               finally
               {
         if (socket != null && !socket.isClosed()) {
            socket.close();
         }
      }
   }
      private static String getCurrentDateTime() {
      SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
      return sdf.format(new Date());
   }
}
```

16. Write a client server program using UDP in which client sends a string to the server andserver replies the strings are palindrome or not.

- **Client side**

```
import java.net.*;
import java.util.Scanner;

class clientpalindrome {
   public static void main(String args[]) {
      try {
         DatagramSocket clientSocket = new DatagramSocket();

         InetAddress serverAddress = InetAddress.getByName("localhost");
         int serverPort = 12345;

         Scanner scanner = new Scanner(System.in);

         System.out.println("Enter a string to check if it's a palindrome:");
         String inputString = scanner.nextLine();

         byte[] sendBuffer = inputString.getBytes();
         DatagramPacket sendPacket = new DatagramPacket(sendBuffer,
sendBuffer.length, serverAddress, serverPort);
         clientSocket.send(sendPacket);

         byte[] receiveBuffer = new byte[1024];
         DatagramPacket receivePacket = new DatagramPacket(receiveBuffer,
receiveBuffer.length);
         clientSocket.receive(receivePacket);

         String result = new String(receivePacket.getData(), 0,
receivePacket.getLength());
         System.out.println("Server response: " + result);

         clientSocket.close();
      }
      catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

- **Server side**

```
import java.net.*;

class serverpalindrome {
   public static void main(String args[]) {
```

```
        try {
            DatagramSocket serverSocket = new DatagramSocket(12345);

            System.out.println("Server is waiting for client messages...");

            while (true) {
                byte[] receiveBuffer = new byte[1024];
                DatagramPacket receivePacket = new
DatagramPacket(receiveBuffer, receiveBuffer.length);

                serverSocket.receive(receivePacket);

                String message = new String(receivePacket.getData(), 0,
receivePacket.getLength());

                String result = isPalindrome(message) ? " string is a Palindrome" :
"string is not a palindrome";

                InetAddress clientAddress = receivePacket.getAddress();
                int clientPort = receivePacket.getPort();
                byte[] sendBuffer = result.getBytes();
                DatagramPacket sendPacket = new DatagramPacket(sendBuffer,
sendBuffer.length, clientAddress, clientPort);
                serverSocket.send(sendPacket);
            }
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static boolean isPalindrome(String str) {
        int left = 0;
        int right = str.length() - 1;

        while (left < right) {
            if (str.charAt(left) != str.charAt(right)) {
                return false;
            }
            left++;
            right--;
        }
        return true;
    }
}
```

17. Write a client server program using UDP in which client sends an integer number to the server and server replies whether it is even number or odd number.

- **Client side**

```
import java.net.*;
import java.util.*;

class clientoddeven {
   public static void main(String args[]) {
      try {
         DatagramSocket clientSocket = new DatagramSocket();
         Scanner scanner = new Scanner(System.in);
         InetAddress serverAddress = InetAddress.getByName("localhost");
         int serverPort = 12345;

         System.out.println("Client is ready to send messages to the
server...");

         while (true) {
            System.out.print("Enter a number (or 'exit' to quit): ");
            String input = scanner.nextLine();

            if (input.equalsIgnoreCase("exit")) {
               break;
            }

            byte[] sendBuffer = input.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendBuffer,
sendBuffer.length, serverAddress, serverPort);
            clientSocket.send(sendPacket);

            byte[] receiveBuffer = new byte[1024];
            DatagramPacket receivePacket = new
DatagramPacket(receiveBuffer, receiveBuffer.length);
            clientSocket.receive(receivePacket);

            String response = new String(receivePacket.getData(), 0,
receivePacket.getLength());
            System.out.println("Server Response: " + response);
         }

         clientSocket.close();
         scanner.close();
      }
            catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

- **Server side**

```
import java.net.*;

class serveroddeven {
   public static void main(String args[]) {
      try {
         DatagramSocket serverSocket = new DatagramSocket(12345);

         System.out.println("Server is waiting for client messages...");

         while (true) {
            byte[] receiveBuffer = new byte[1024];
            DatagramPacket receivePacket = new
DatagramPacket(receiveBuffer, receiveBuffer.length);

            serverSocket.receive(receivePacket);

            String message = new String(receivePacket.getData(), 0,
receivePacket.getLength()).trim();

            try {
               int number = Integer.parseInt(message);
               String result = isEven(number) ? number + " no is even " :
number + " no is odd ";

               InetAddress clientAddress = receivePacket.getAddress();
               int clientPort = receivePacket.getPort();
               byte[] sendBuffer = result.getBytes();
               DatagramPacket sendPacket = new
DatagramPacket(sendBuffer, sendBuffer.length, clientAddress, clientPort);
               serverSocket.send(sendPacket);
            }
                              catch (NumberFormatException e) {
               String errorMessage = "Invalid number received!";
               InetAddress clientAddress = receivePacket.getAddress();
               int clientPort = receivePacket.getPort();
               byte[] sendBuffer = errorMessage.getBytes();
               DatagramPacket sendPacket = new
DatagramPacket(sendBuffer, sendBuffer.length, clientAddress, clientPort);
               serverSocket.send(sendPacket);
            }
         }
      }
               catch (Exception e) {
         e.printStackTrace();
      }
   }
```

```
    public static boolean isEven(int number) {
        return number % 2 == 0;
    }
}
```

18. Write a client server program using UDP in which client sends an integer number to theserver and server replies whether it is prime number or not.

- **Client side**

```
import java.net.*;
import java.util.*;

class clientprime {
    public static void main(String args[]) {
        try {
            DatagramSocket clientSocket = new DatagramSocket();
            Scanner scanner = new Scanner(System.in);
            InetAddress serverAddress = InetAddress.getByName("localhost");
            int serverPort = 12345;

            System.out.println("Client is ready to send messages to the
server...");

            while (true) {
                System.out.print("Enter a number (or 'exit' to quit): ");
                String input = scanner.nextLine();

                if (input.equalsIgnoreCase("exit")) {
                    break;
                }

                byte[] sendBuffer = input.getBytes();
                DatagramPacket sendPacket = new DatagramPacket(sendBuffer,
sendBuffer.length, serverAddress, serverPort);
                clientSocket.send(sendPacket);

                byte[] receiveBuffer = new byte[1024];
                DatagramPacket receivePacket = new
DatagramPacket(receiveBuffer, receiveBuffer.length);
                clientSocket.receive(receivePacket);

                String response = new String(receivePacket.getData(), 0,
receivePacket.getLength());
                System.out.println("Server Response: " + response);
            }
            clientSocket.close();
            scanner.close();
```

```
        }
                catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

- **Server side**

```
import java.net.*;

class serverprime {
    public static void main(String args[]) {
        try {
            DatagramSocket serverSocket = new DatagramSocket(12345);

            System.out.println("Server is waiting for client messages...");

            while (true) {
                byte[] receiveBuffer = new byte[1024];
                DatagramPacket receivePacket = new
DatagramPacket(receiveBuffer, receiveBuffer.length);

                serverSocket.receive(receivePacket);

                String message = new String(receivePacket.getData(), 0,
receivePacket.getLength()).trim();

                try {
                    int number = Integer.parseInt(message);
                    String result = isPrime(number) ? number + " no is Prime " :
number + " no is Not Prime ";

                    InetAddress clientAddress = receivePacket.getAddress();
                    int clientPort = receivePacket.getPort();
                    byte[] sendBuffer = result.getBytes();
                    DatagramPacket sendPacket = new
DatagramPacket(sendBuffer, sendBuffer.length, clientAddress, clientPort);
                    serverSocket.send(sendPacket);
                }
                                catch (NumberFormatException e) {
                    String errorMessage = "Invalid number received!";
                    InetAddress clientAddress = receivePacket.getAddress();
                    int clientPort = receivePacket.getPort();
                    byte[] sendBuffer = errorMessage.getBytes();
                    DatagramPacket sendPacket = new
DatagramPacket(sendBuffer, sendBuffer.length, clientAddress, clientPort);
                    serverSocket.send(sendPacket);
                }
```

```
            }
        }
                catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static boolean isPrime(int number) {
        if (number <= 1) {
            return false;
        }
        for (int i = 2; i <= Math.sqrt(number); i++) {
            if (number % i == 0) {
                return false;
            }
        }
        return true;
    }
}
```

19. Write a client server program using UDP in which client chats with the server. It should be a two-way chat.

- **Client side**

```
import java.net.*;
import java.io.*;
class chatclient {
    public static void main(String args[]) {
        DatagramSocket socket = null;

        try {
            socket = new DatagramSocket();
            InetAddress serverAddress = InetAddress.getByName("localhost");
            int serverPort = 9876;

            BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));

            while (true) {
                System.out.print("Client: ");
                String message = reader.readLine();

                byte[] sendData = message.getBytes();
                DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, serverAddress, serverPort);
                socket.send(sendPacket);

                if (message.equalsIgnoreCase("exit")) {
```

```
                System.out.println("Chat ended by client.");
                break;
            }

            byte[] receiveData = new byte[1024];
            DatagramPacket receivePacket = new
DatagramPacket(receiveData, receiveData.length);
            socket.receive(receivePacket);

            String serverResponse = new String(receivePacket.getData(), 0,
receivePacket.getLength());
            System.out.println("Server: " + serverResponse);

            if (serverResponse.equalsIgnoreCase("exit")) {
                System.out.println("Chat ended by server.");
                break;
            }
        }
    }
            catch (Exception e)
            {
        e.printStackTrace();
    }
            finally
            {
        if (socket != null && !socket.isClosed()) {
            socket.close();
        }
    }
  }
}
```

- **Server side**

```
import java.net.*;
import java.io.*;

class chatserver {
   public static void main(String args[]) {
      DatagramSocket socket = null;

      try {
         socket = new DatagramSocket(9876);
         System.out.println("Server is ready to chat...");

         while (true) {
            byte[] receiveData = new byte[1024];
            DatagramPacket receivePacket = new
DatagramPacket(receiveData, receiveData.length);
```

```
            socket.receive(receivePacket);

            String clientMessage = new String(receivePacket.getData(), 0,
receivePacket.getLength());
            System.out.println("Client: " + clientMessage);

            if (clientMessage.equalsIgnoreCase("exit")) {
                System.out.println("Chat ended by client.");
                break;
            }

            BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));
            System.out.print("Server: ");
            String serverMessage = reader.readLine();

            byte[] sendData = serverMessage.getBytes();
            InetAddress clientAddress = receivePacket.getAddress();
            int clientPort = receivePacket.getPort();

            DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, clientAddress, clientPort);
            socket.send(sendPacket);

            if (serverMessage.equalsIgnoreCase("exit")) {
                System.out.println("Chat ended by server.");
                break;
            }
        }
    }
              catch (Exception e)
              {
        e.printStackTrace();
    }
              finally
              {
        if (socket != null && !socket.isClosed()) {
            socket.close();
        }
    }
}
```