Dr. M.G.R.
EDUCATIONAL AND RESEARCH INSTITUTE
DEEMED TO BE UNIVERSITY
University with Graded Autonomy Status
(An ISO 21001 : 2018 Certified Institution)
Periyar E.V.R. High Road, Maduravoyal, Chennai-95. Tamilnadu, India.

RECORD NOTEBOOK

BCS18L05-NETWORK PROGRAMMING LAB

DEPARTMENT
OF
COMPUTER SCIENCE AND ENGINEERING

NAME	**:**	HARICHSELVAM
REGISTER NO	**:**	211191101045
COURSE	**:**	B. TECH CSE(DS&AI)
YEAR/SEM/SEC	**:**	III/V/A

2023-2024 (ODD SEMESTER)

Dr. M.G.R.
EDUCATIONAL AND RESEARCH INSTITUTE
DEEMED TO BE UNIVERSITY
University with Graded Autonomy Status
(An ISO 21001 : 2018 Certified Institution)
Periyar E.V.R. High Road, Maduravoyal, Chennai-95. Tamilnadu, India.

BONAFIDE CERTIFICATE

REGISTER NO: 211191101045

NAME OF LAB: NETWORK PROGRAMMING LAB –(BCS18L05)

DEPARTMENT: COMPUTER SCIENCE AND ENGINEERING

Certified that, this Record note book is a bonafide record of work done by HARICHSELVAM of III Year B. Tech CSE(DS&AI), Sec- 'A' in the network programming lab during the year 2023-2024.

Signature of Lab-in-Charge **Signature of Head of Dept**

Submitted for the Practical Examination held on ----------

Internal Examiner **External Examiner**

INDEX

Exp No	Date	Name Of Experiment	Page No	Staff Signature
1		Networking Commands with Options	1	
2		Socket program to extent communication between two deferent ends using TCP	15	
3		Socket program to extent communication between two deferent ends using UDP	22	
4		Create a Socket (TCP) between two computers and enable file transfer between them.	27	
5		Implementation of RPC in server-client model	32	
6		Implementation of ARP/RARP	38	
7		HTTP Socket program to download a web page	43	
8		File transfer in Client-Server architecture using following methods a) Using RS232C b) Using TCP/IP	47	
9		To implement RMI (Remote Method Invocation)	63	
10		Write a network program to broadcast/ multicast a message to a group in the samenetwork.	68	

Ex. No:1 **Date:**

NETWORKING COMMANDS WITH OPTIONS

AIM:

To work on networking commands with options in Windows Operating System.

ALGORITHM:

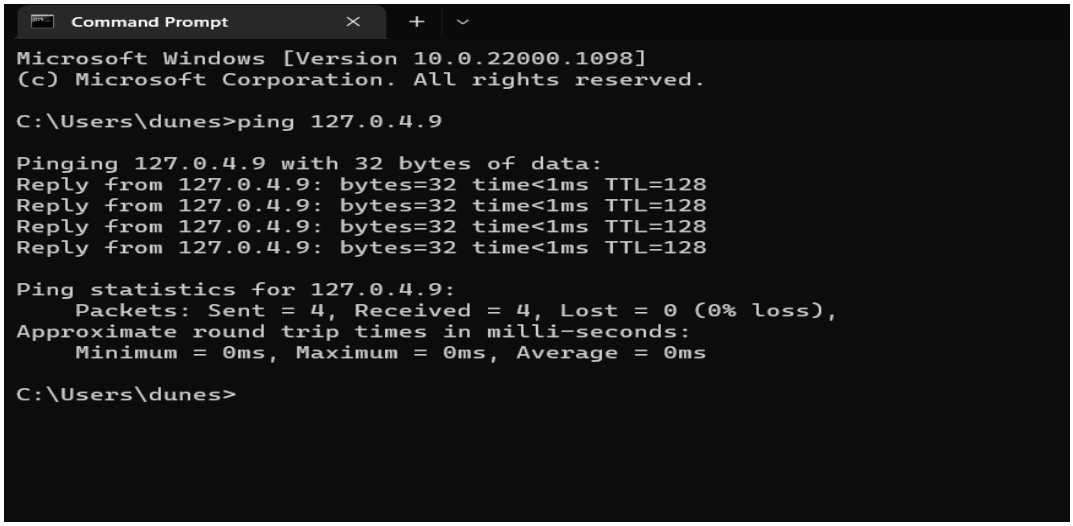
1. Start the program.
2. Open the command prompt
3. Enter the commands along with proper options
4. View the Command Output

PROGRAM:

PING:

EXPLANATION: This is used to provide a basic connectivity test between the requesting host and destination host**.**

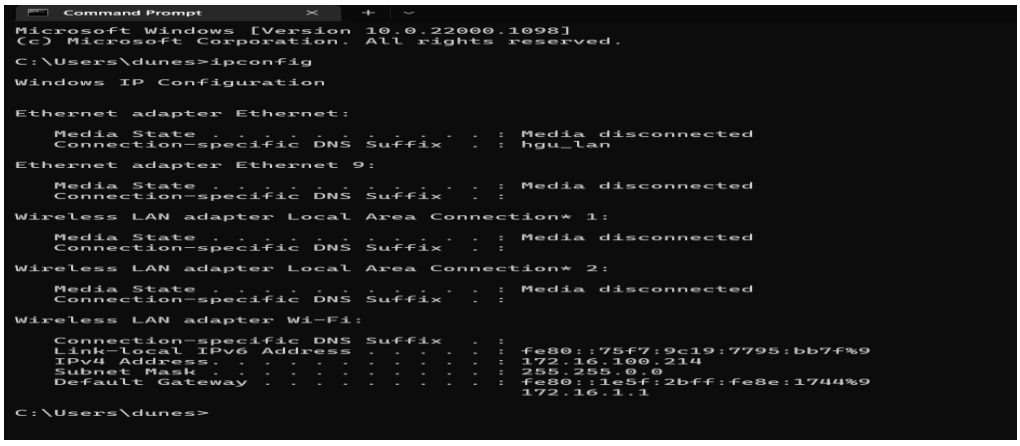
OUTPUT:



IPCONFIG:

ipconfig [/allcompartments] [/? | /all | /renew [adapter] | /release [adapter] | /renew6 [adapter] | /release6 [adapter] | /flushdns | /displaydns | /registerdns | /showclassid adapter | /setclassid adapter [classid] | /showclassid6 adapter | /setclassid6 adapter [classid]]

OUTPUT:

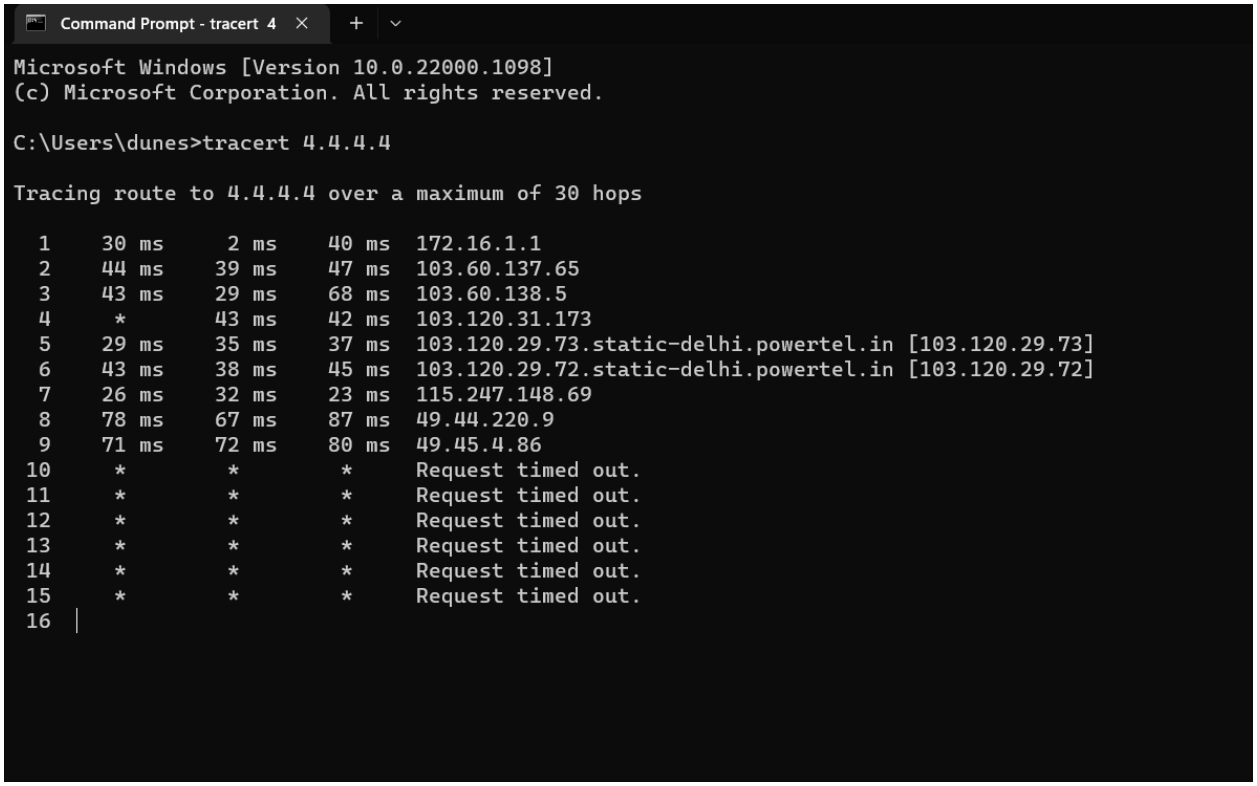


TRACERT:

tracert [-d] [-h maximum_hops] [-j host-list] [-w timeout] [-R] [-S srcaddr] [-4] [-6]

target_name

OUTPUT:

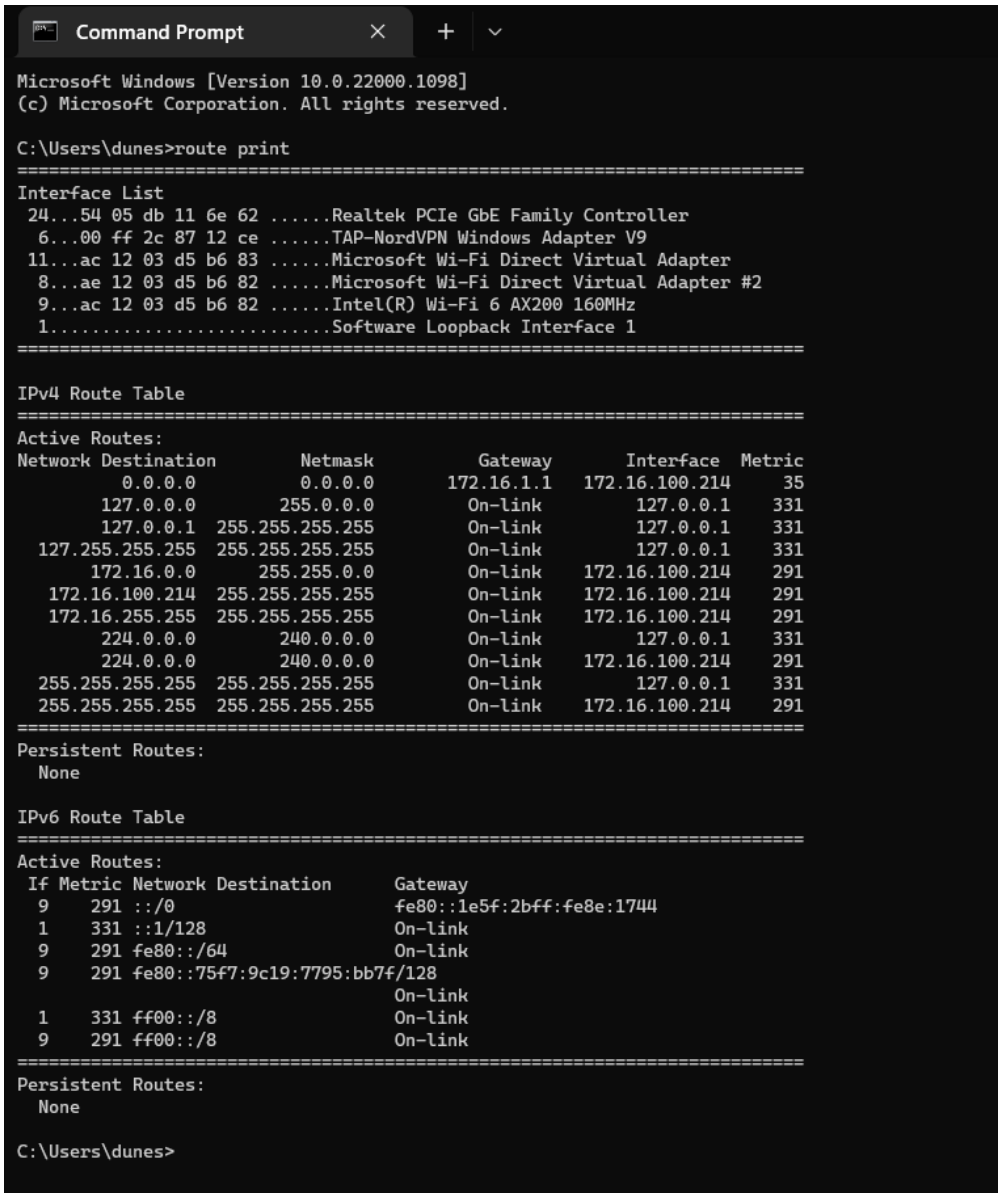


ROUTE:

ROUTE [-f] [-p] [-4|-6] command [destination] [MASK netmask] [gateway] [METRIC

metric] [IF interface]

OUTPUT:



NBTSTAT:

NBTSTAT [[-a RemoteName] [-A IP address] [-c] [-n] [-r] [-R] [-RR] [-s] [-S] [interval]]

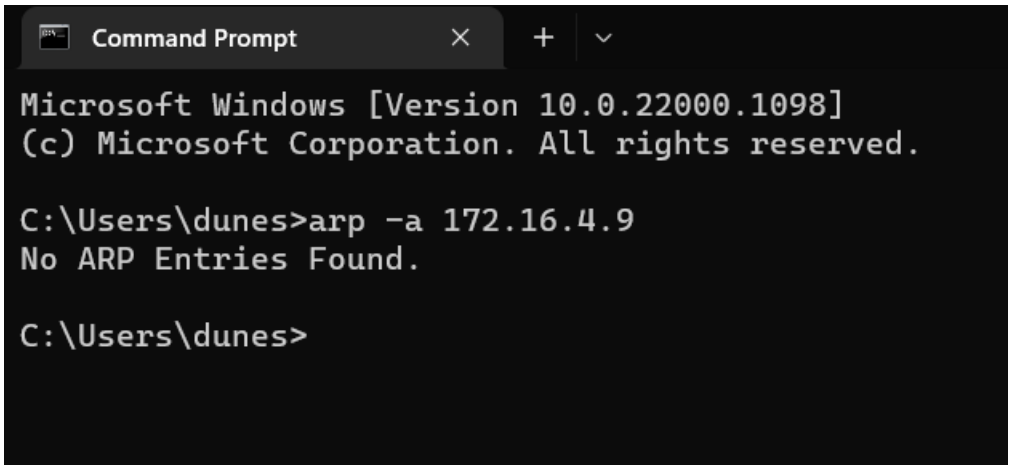
OUTPUT:

```
C:\Users\dunes>nbtstat /n

Ethernet 9:
Node IpAddress: [0.0.0.0] Scope Id: []

    No names in cache

Ethernet:
Node IpAddress: [0.0.0.0] Scope Id: []

    No names in cache

Wi-Fi:
Node IpAddress: [172.16.100.214] Scope Id: []

                NetBIOS Local Name Table

       Name               Type         Status
    ---------------------------------------------
    ATTICUS        <20>  UNIQUE      Registered
    ATTICUS        <00>  UNIQUE      Registered
    WORKGROUP      <00>  GROUP       Registered

Local Area Connection* 1:
Node IpAddress: [0.0.0.0] Scope Id: []

    No names in cache

Local Area Connection* 2:
Node IpAddress: [0.0.0.0] Scope Id: []

                NetBIOS Local Name Table

       Name               Type         Status
    ---------------------------------------------
    ATTICUS        <20>  UNIQUE      Registered

Bluetooth Network Connection:
Node IpAddress: [0.0.0.0] Scope Id: []

    No names in cache

C:\Users\dunes>
```

ARP:

EXPLANATION: It stands for address resolution protocol. Used to display and modify entries in address resolution protocol cache

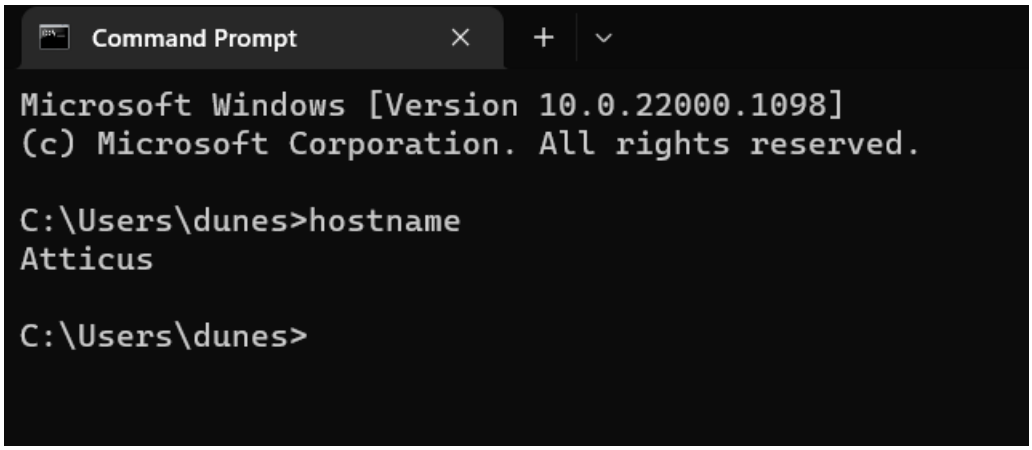
OUTPUT:



HOSTNAME:

EXPLANATION: Hostname this is used to return local computer name.

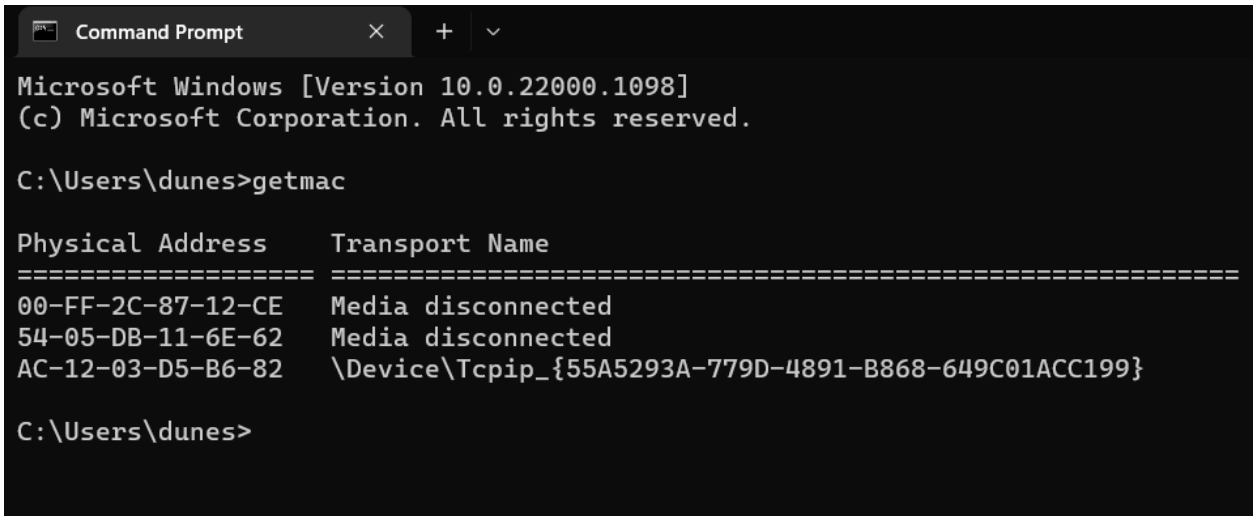
OUTPUT:



GETMAC:

EXPLANATION: Getmac used to show both local and mac address.

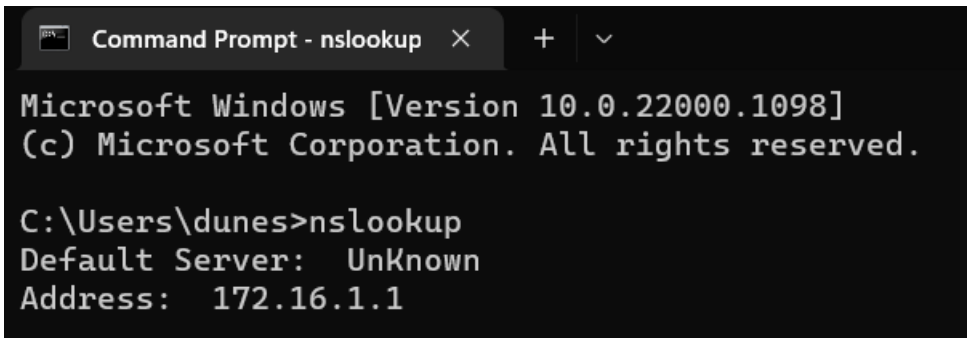
OUTPUT:



NSLOOKUP:

EXPLANATION: Nslookup this utility used to lookup specific ip address associated with domain name.

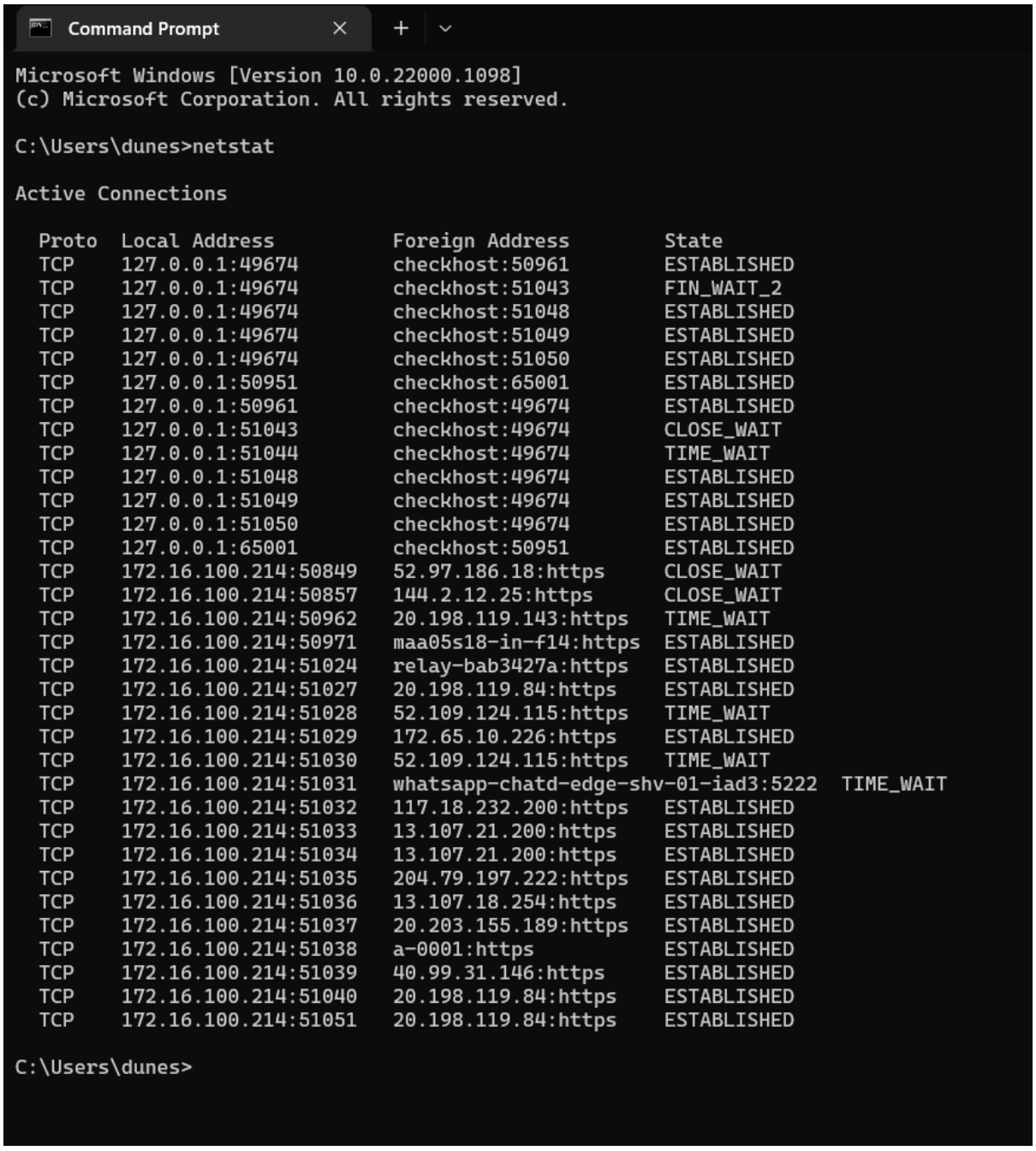
OUTPUT:



NETSAT:

EXPLANATION: Netstat this is used to figure out correct state of active network connection on a host.

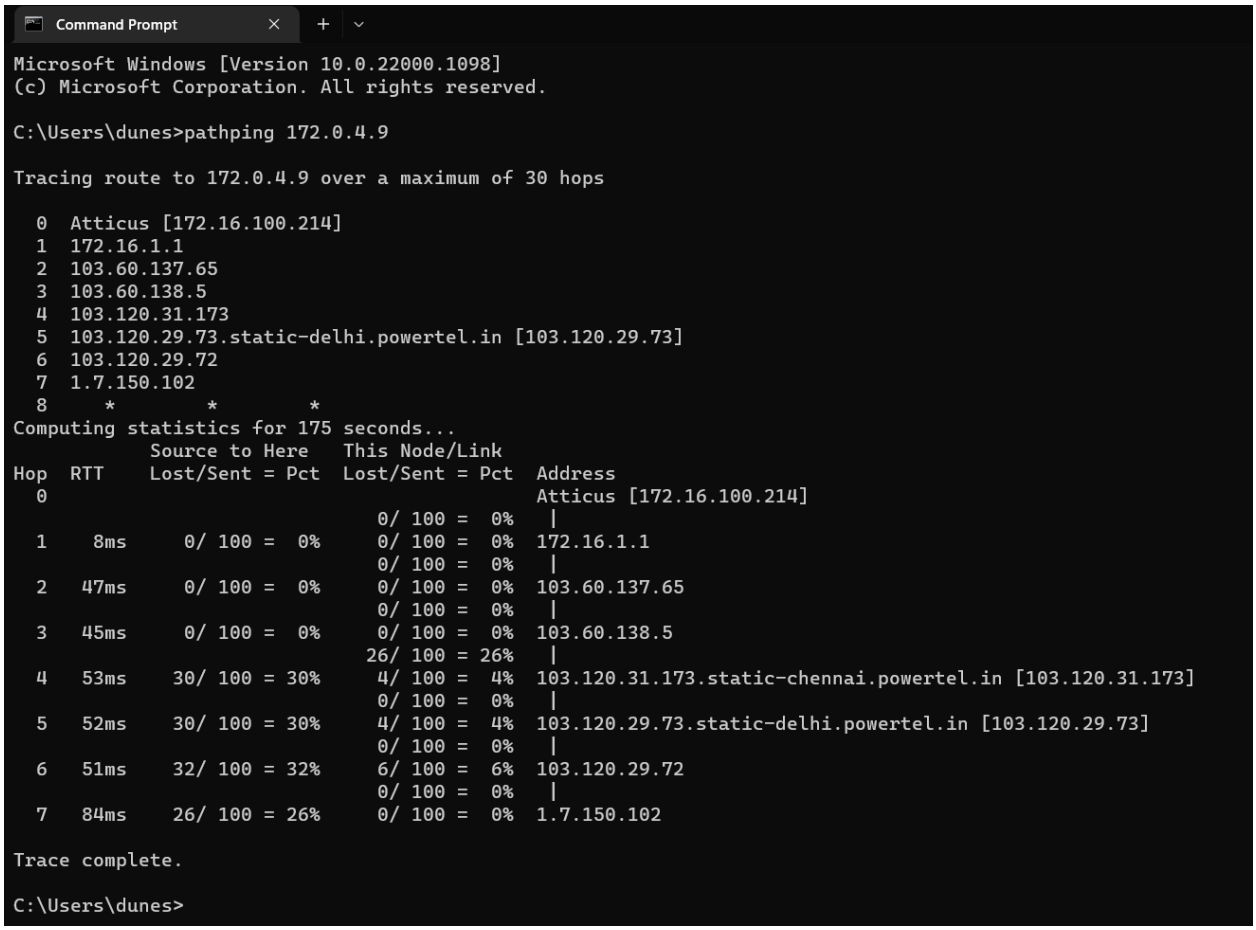
OUTPUT:



PATHPING:

EXPLANATION: Pathping it take the functionality and information that can be obtained from this type of tools provides information about network latency and network loss of intermediate hops between a source and destination.

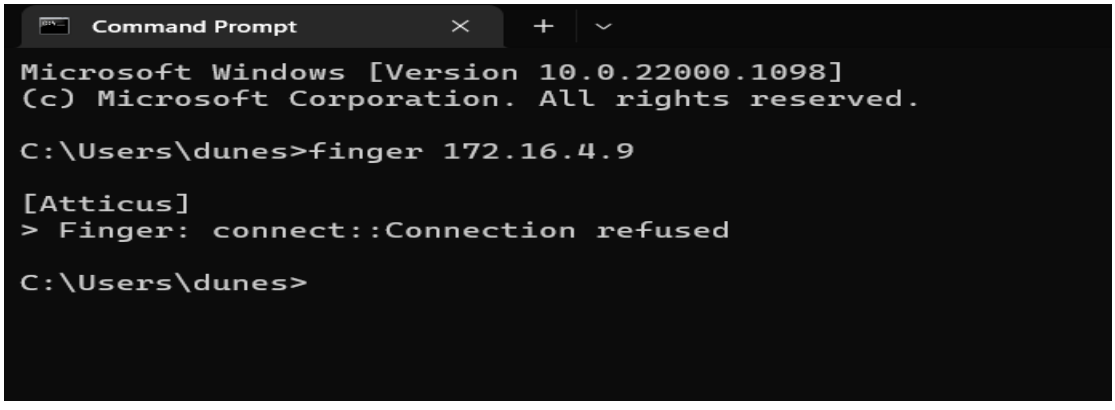
OUTPUT:



FINGER:

EXPLANATION: Finger this enables a user to view other user information when user in the same system.

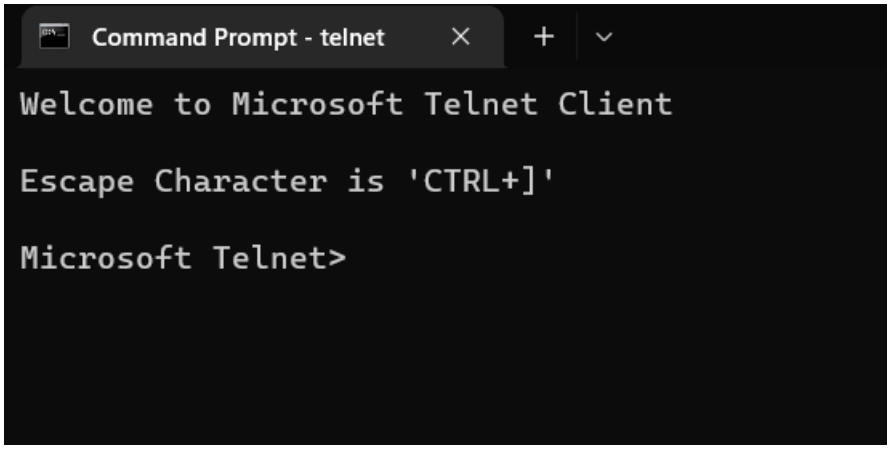
OUTPUT:



TELNET:

EXPLANATION: this is a software that allow user to remotely access another computer such as server etc..

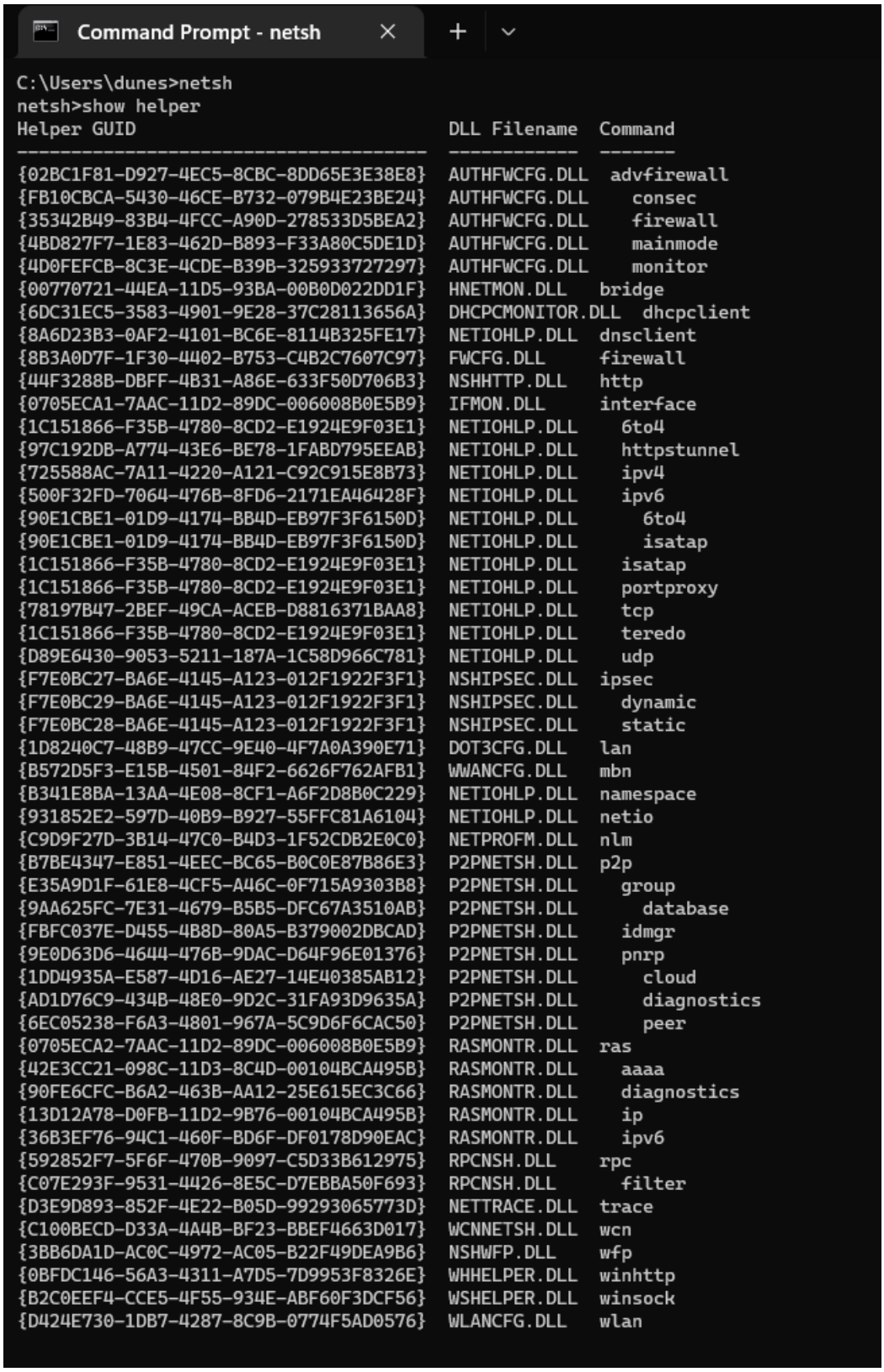
OUTPUT:



NET SH:

EXPLANATION: Netsh used to display protocol statistics and current tcp/ip connection using nbt.

OUTPUT:

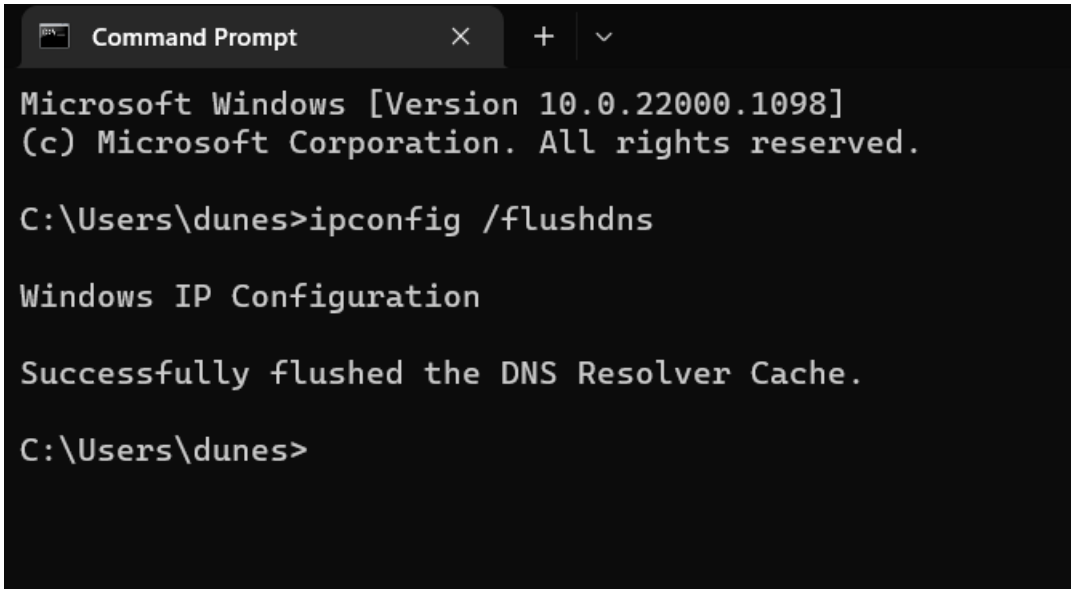

Command Prompt - netsh
C:\Users\dunes>netsh
netsh>show helper
Helper GUID DLL Filename Command
{02BC1F81-D927-4EC5-8CBC-8DD65E3E38E8} AUTHFWCFG.DLL advfirewall
{FB10CBCA-5430-46CE-B732-079B4E23BE24} AUTHFWCFG.DLL consec
{35342B49-83B4-4FCC-A90D-278533D5BEA2} AUTHFWCFG.DLL firewall
{4BD827F7-1E83-462D-B893-F33A80C5DE1D} AUTHFWCFG.DLL mainmode
{4D0FEFCB-8C3E-4CDE-B39B-325933727297} AUTHFWCFG.DLL monitor
{00770721-44EA-11D5-93BA-00B0D022DD1F} HNETMON.DLL bridge
{6DC31EC5-3583-4901-9E28-37C28113656A} DHCPCMONITOR.DLL dhcpclient
{8A6D23B3-0AF2-4101-BC6E-8114B325FE17} NETIOHLP.DLL dnsclient
{8B3A0D7F-1F30-4402-B753-C4B2C7607C97} FWCFG.DLL firewall
{44F3288B-DBFF-4B31-A86E-633F50D706B3} NSHHTTP.DLL http
{0705ECA1-7AAC-11D2-89DC-006008B0E5B9} IFMON.DLL interface
{1C151866-F35B-4780-8CD2-E1924E9F03E1} NETIOHLP.DLL 6to4
{97C192DB-A774-43E6-BE78-1FABD795EEAB} NETIOHLP.DLL httpstunnel
{725588AC-7A11-4220-A121-C92C915E8B73} NETIOHLP.DLL ipv4
{500F32FD-7064-476B-8FD6-2171EA46428F} NETIOHLP.DLL ipv6
{90E1CBE1-01D9-4174-BB4D-EB97F3F6150D} NETIOHLP.DLL 6to4
{90E1CBE1-01D9-4174-BB4D-EB97F3F6150D} NETIOHLP.DLL isatap
{1C151866-F35B-4780-8CD2-E1924E9F03E1} NETIOHLP.DLL isatap
{1C151866-F35B-4780-8CD2-E1924E9F03E1} NETIOHLP.DLL portproxy
{78197B47-2BEF-49CA-ACEB-D8816371BAA8} NETIOHLP.DLL tcp
{1C151866-F35B-4780-8CD2-E1924E9F03E1} NETIOHLP.DLL teredo
{D89E6430-9053-5211-187A-1C58D966C781} NETIOHLP.DLL udp
{F7E0BC27-BA6E-4145-A123-012F1922F3F1} NSHIPSEC.DLL ipsec
{F7E0BC29-BA6E-4145-A123-012F1922F3F1} NSHIPSEC.DLL dynamic
{F7E0BC28-BA6E-4145-A123-012F1922F3F1} NSHIPSEC.DLL static
{1D8240C7-48B9-47CC-9E40-4F7A0A390E71} DOT3CFG.DLL lan
{B572D5F3-E15B-4501-84F2-6626F762AFB1} WWANCFG.DLL mbn
{B341E8BA-13AA-4E08-8CF1-A6F2D8B0C229} NETIOHLP.DLL namespace
{931852E2-597D-40B9-B927-55FFC81A6104} NETIOHLP.DLL netio
{C9D9F27D-3B14-47C0-B4D3-1F52CDB2E0C0} NETPROFM.DLL nlm
{B7BE4347-E851-4EEC-BC65-B0C0E87B86E3} P2PNETSH.DLL p2p
{E35A9D1F-61E8-4CF5-A46C-0F715A9303B8} P2PNETSH.DLL group
{9AA625FC-7E31-4679-B5B5-DFC67A3510AB} P2PNETSH.DLL database
{FBFC037E-D455-4B8D-80A5-B379002DBCAD} P2PNETSH.DLL idmgr
{9E0D63D6-4644-476B-9DAC-D64F96E01376} P2PNETSH.DLL pnrp
{1DD4935A-E587-4D16-AE27-14E40385AB12} P2PNETSH.DLL cloud
{AD1D76C9-434B-48E0-9D2C-31FA93D9635A} P2PNETSH.DLL diagnostics
{6EC05238-F6A3-4801-967A-5C9D6F6CAC50} P2PNETSH.DLL peer
{0705ECA2-7AAC-11D2-89DC-006008B0E5B9} RASMONTR.DLL ras
{42E3CC21-098C-11D3-8C4D-00104BCA495B} RASMONTR.DLL aaaa
{90FE6CFC-B6A2-463B-AA12-25E615EC3C66} RASMONTR.DLL diagnostics
{13D12A78-D0FB-11D2-9B76-00104BCA495B} RASMONTR.DLL ip
{36B3EF76-94C1-460F-BD6F-DF0178D90EAC} RASMONTR.DLL ipv6
{592852F7-5F6F-470B-9097-C5D33B612975} RPCNSH.DLL rpc
{C07E293F-9531-4426-8E5C-D7EBBA50F693} RPCNSH.DLL filter
{D3E9D893-852F-4E22-B05D-99293065773D} NETTRACE.DLL trace
{C100BECD-D33A-4A4B-BF23-BBEF4663D017} WCNNETSH.DLL wcn
{3BB6DA1D-AC0C-4972-AC05-B22F49DEA9B6} NSHWFP.DLL wfp
{0BFDC146-56A3-4311-A7D5-7D9953F8326E} WHHELPER.DLL winhttp
{B2C0EEF4-CCE5-4F55-934E-ABF60F3DCF56} WSHELPER.DLL winsock
{D424E730-1DB7-4287-8C9B-0774F5AD0576} WLANCFG.DLL wlan

IPCONFIG/FLUSHDNS:

EXPLANATION: Ipconfig flushdns it displays all current TCP/IP network configuration

values and reference DHCP.

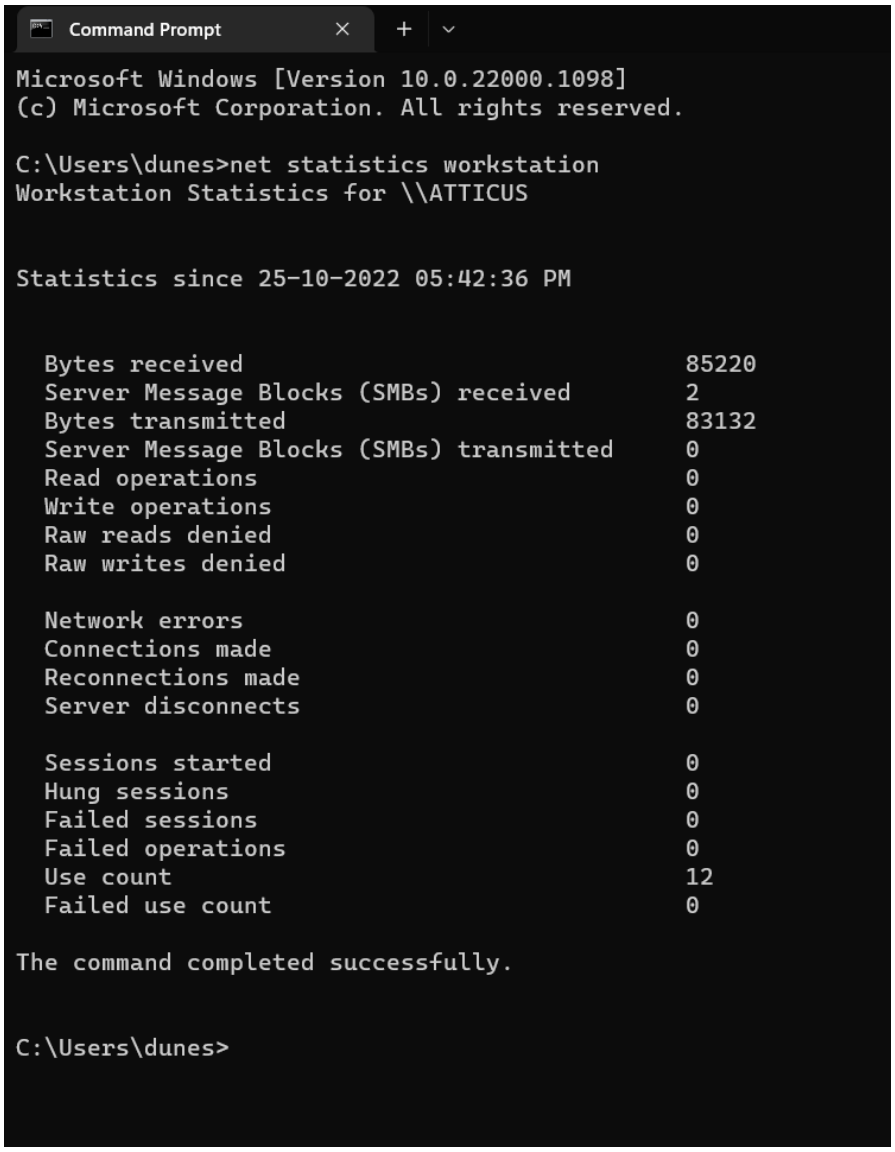
OUTPUT:



NET:

EXPLANATION: net used to create, remove and otherwise manage shared resources on system.

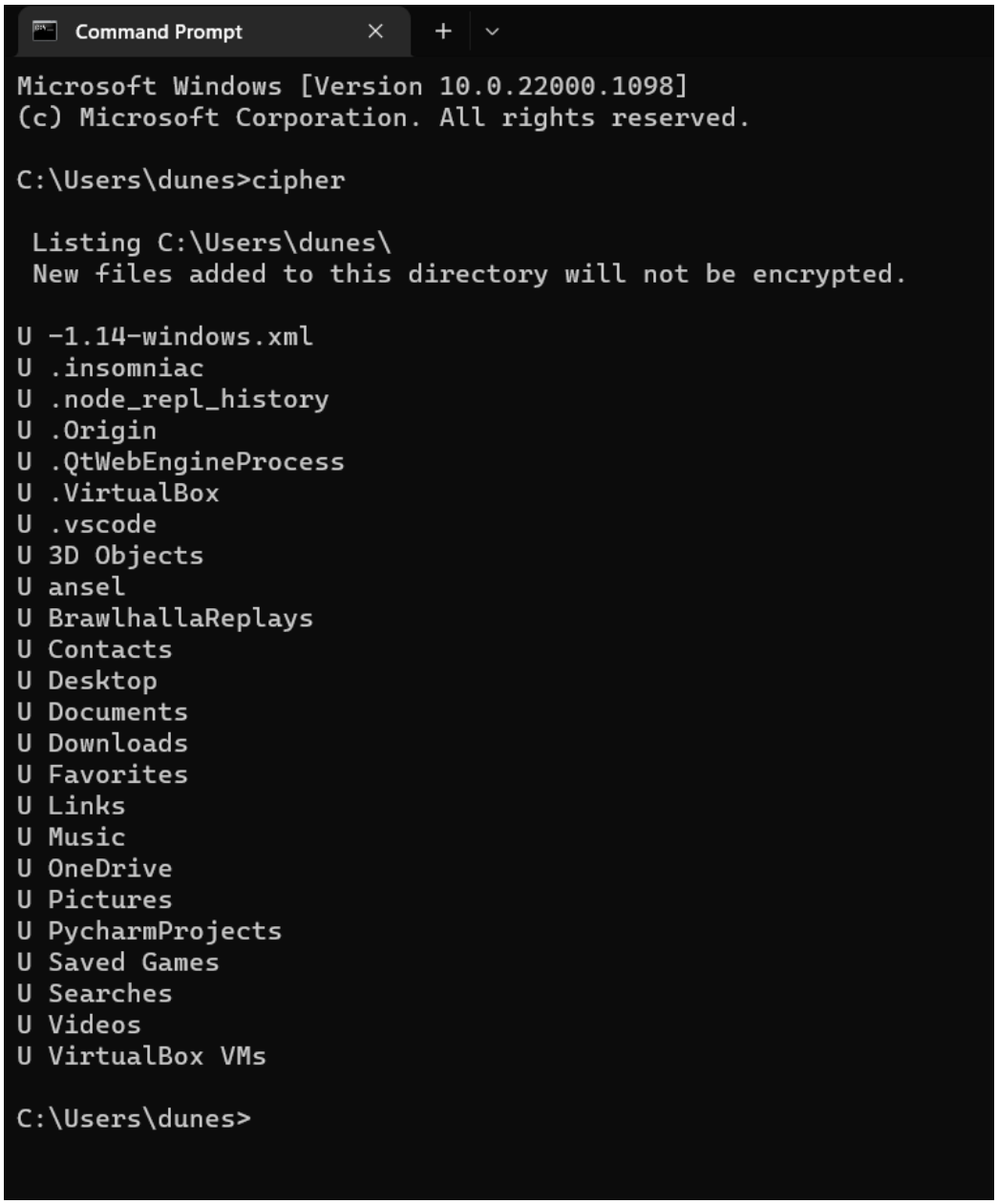
OUTPUT:



CIPHER:

EXPLANATION: Cipher this is a built-in command line tool in os used to encrypt or decrypt data on ntfs drives.

OUTPUT:



RESULT:

Thus, the Network Commands with options for Windows Operating System has been executed successfully

Ex.No:2 **Date:**

SOCKET PROGRAM TO EXTENT COMMUNICATION BETWEEN TWO DIFFERENT ENDS USING TCP

AIM:

To write a java program for implementing Socket program to extent communication between two deferent ends using TCP**.**

ALGORITHM:

SERVER

Step 1: Start the program.

Step 2: Create an unnamed socket for the server using the

parameters AF_INET as domain and the SOCK_STREAM as type.

Step 3: Name the socket using bind( ) system call with the

parameters server_sockfd and the server address(sin_addr and sin_sport).

Step 4: Create a connection queue and wait for clients using the listen( ) system call

with the number of clients request as parameters.

Step 5: Accept the connection using accept( ) system call when client requests for connection.

Step 6: Get the message which has to be sent to the client and check that it is not equal to ‘Bye’.

Step 7: If the message is not equal to ‘Bye’ then write the message to the client and Goto step 6.

Step 8: If the message is ‘Bye’ then terminate the Process.

Step 9: Stop the program execution.

CLIENT

Step 1: Start the program.

Step 2: Create an unnamed socket for client using socket( ) system.

Step 3: Call with parameters AF_INET as domain and SOCK_STREAM as type.

Step 4: Name the socket using bind( ) system call.

Step 5: Now connect the socket to server using connect( ) system call.

Step 6: Read the message from the server socket and compare it with ‘Bye’.

Step 7: If the message is not equal to ‘Bye’ then print the message to the server output device and repeat the steps 6 & 7.

Step 8: Get the message from the client side.

Step 9: Write the message to server sockfd and goto step 4.

Step 10: If the message is equal to ‘Bye’then print good bye message and terminate the process.

Step 11: Stop the process.

PROGRAM:

SERVER

```
import java.net.*;

import java.io.*;

public class Server

{

private Socket socket = null;

private ServerSocket server = null;

private DataInputStream in = null;

public Server(int port)

{

try{

server = new ServerSocket(port);

System.out.println("This program is done by HARICHSELVAM 211191101045");

System.out.println("Server started");

System.out.println("Waiting for a client ...");

socket = server.accept();

System.out.println("Client accepted");

in = new DataInputStream(

new BufferedInputStream(socket.getInputStream()));

String line = "";

while (!line.equals("Over"))
```

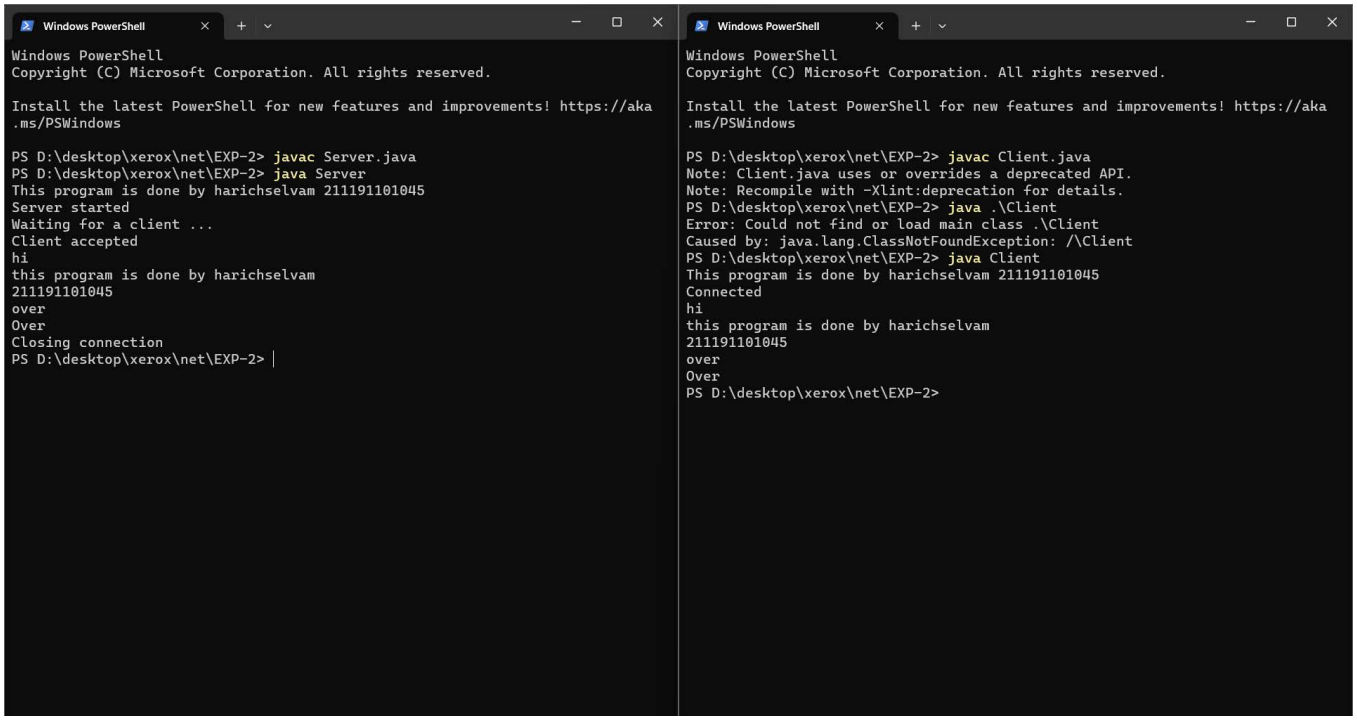
```
{
try{
line = in.readUTF();
System.out.println(line);
}
catch(IOException i)
{
System.out.println(i);
}}
System.out.println("Closing connection");
socket.close();
in.close();
}
catch(IOException i)
{
System.out.println(i);
}}
public static void main(String args[])
{
Server server = new Server(5000);
}
}
```

CLIENT

```
import java.net.*;

import java.io.*;

public class Client

{

private Socket socket = null;

private DataInputStream input = null;

private DataOutputStream out = null;

public Client(String address, int port)

{

try{

socket = new Socket(address, port);

System.out.println("This program is done by HARICHSELVAM 211191101045");

System.out.println("Connected");

input = new DataInputStream(System.in);

out = new DataOutputStream(socket.getOutputStream());

}

catch(UnknownHostException u)

{

System.out.println(u);}

catch(IOException i){

System.out.println(i);
```

```
}
String line = "";
while (!line.equals("Over"))
{
try{
line = input.readLine();
out.writeUTF(line);}
catch(IOException i){
System.out.println(i);
}}
try{
input.close();
out.close();
socket.close();
}
catch(IOException i){
System.out.println(i);
}}
public static void main(String args[]){
Client client = new Client("127.0.0.1", 5000);
}
}
```

OUTPUT:



RESULT:

Thus, the java program for implementing Socket program to extent communication between two deferent ends using TCP is executed and the output is verified successfully**.**

Ex.No:3 **Date:**

SOCKET PROGRAM TO EXTENT COMMUNICATION BETWEEN TWO DEFERENT ENDS USING UDP

AIM: To write a java program for implementing Socket program to extent communication between two deferent ends using UDP.

ALGORITHM:

SERVER

Step 1: Start the program.

Step 2: Create an unnamed socket for the server using the parameters AF_INET as domain and the SOCK_DGRAM as type.

Step 3: Name the socket using bind( ) system call with the parameters server sock and the server address(sin_addr and sin_sport).

Step 4: The server gets the message from the client.

Step 5: Prints the message.

Step 6: Stop the program execution.

CLIENT

Step 1: Start the program.

Step 2: Create an unnamed socket for client using socket

Step 3: Call with parameters AF_INET as domain an SOCK_DGRAM as type.

Step 4: Name the socket using bind( ) system call.

Step 5: The Sendto( ) system call is used to deliver the Message to the server.

Step 6: Stop the program execution.

PROGRAM:

SERVER

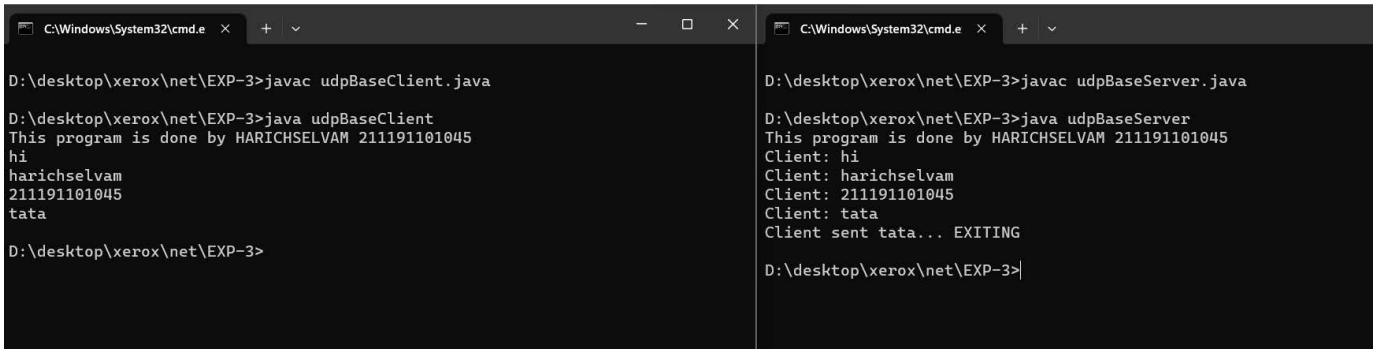
```
import java.io.IOException;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

import java.net.SocketException;

public class udpBaseServer

{

public static void main(String[] args) throws IOException

{DatagramSocket ds = new DatagramSocket(1234);

byte[] receive = new byte[65535];

DatagramPacket DpReceive = null;

System.out.println("This program is done by HARICHSELVAM 211191101045"); while

(true)

{

DpReceive = new DatagramPacket(receive, receive.length);

ds.receive(DpReceive);

System.out.println("Client:-" + data(receive));

if (data(receive).toString().equals("tata"))

{

System.out.println("Client sent bye.....EXITING");
```

```
break;
}
receive = new byte[65535];
}
}
public static StringBuilder data(byte[] a)
{
if (a == null)
return null;
StringBuilder ret = new StringBuilder();
int i = 0;
while (a[i] != 0)
{
ret.append((char) a[i]);
i++;
}
return ret;
}
}
```

CLIENT

```
import java.io.IOException;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

import java.util.Scanner;

public class udpBaseClient

{

public static void main(String args[]) throws IOException{

System.out.println("This program is done by HARICHSELVAM 211191101045"); Scanner

sc = new Scanner(System.in);

DatagramSocket ds = new DatagramSocket();

InetAddress ip = InetAddress.getLocalHost();

byte buf[] = null;

while (true){

String inp = sc.nextLine();

buf = inp.getBytes();

DatagramPacket DpSend = new DatagramPacket(buf, buf.length, ip, 1234);

ds.send(DpSend);

if (inp.equals("tata"))

break;

}}}
```

OUTPUT:



RESULT:

Thus, the given program for the connection between the client and server has been established and successfully verified using UDP.

Ex.No:4 **Date:**

CREATE A SOCKET (TCP) BETWEEN TWO COMPUTERS AND ENABLE FILE TRANSFER BETWEEN THEM

AIM:

To write a java program for transferring a file between two computers using TCP.

ALGORITHM:

SERVER:

Step 1: Start the program.

Step 2: Create an unnamed socket for the server using parameters AF_INET as domain and SOCK_STREAM as type.

Step 3: Get the server port number.

Step 4: Register the host address to the system by using bind () system call in server side.

Step 5: Create a connection queue and wait for clients using listen () system call with the number of clients requests as parameter.

Step 6: Create a Child process using fork () system call.

Step 7: If the process identification number is equal to zero accept the connection using accept () system call when the client request for connection.

Step 8: Ifpid is not equal to zero then exit the process.

Step 9: Stop the Program execution.

CLIENT:

Step 1: Start the program.

Step 2:Create an unnamed socket for the client using parameters AF_INET as

domain and SOCK_STREAM as type.

Step 3:Get the client port number.

Step 4:Now connect the socket to server using connect( ) system call.

Step 5:Enter the file name.

Step 6:The file is transferred from client to server using send ( ) function.

Step 7:Print the contents of the file in a new file.

Step 8:Stop the program.

PROGRAM:

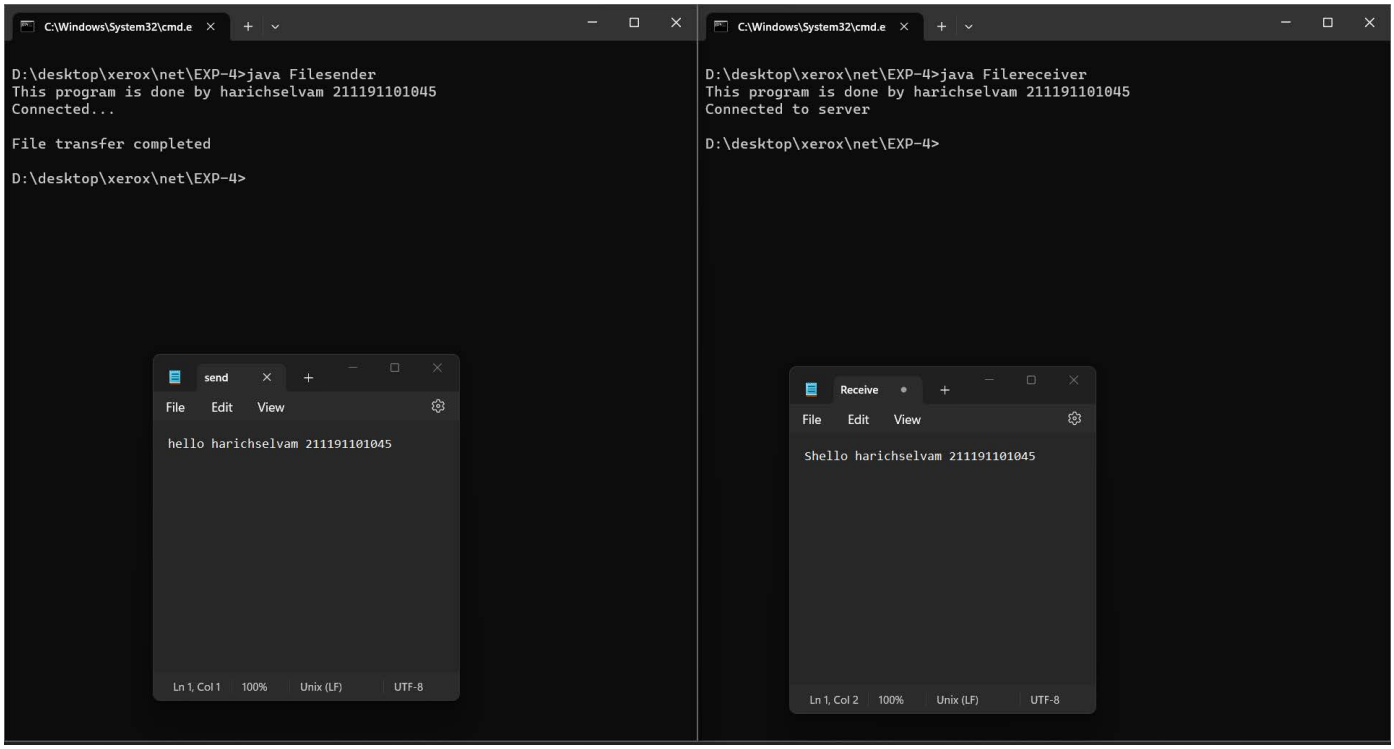
SERVER

```
import java.io.*;

import java.net.*;

class Filesender

{

public static void main(String args[]) throws IOException

{

ServerSocket ss=new ServerSocket(7777);

Socket s=ss.accept();

System.out.println("This program is done by HARICHSELVAM 211191101045");

System.out.println("connected...");

FileInputStream fin=new FileInputStream("send.txt");

DataOutputStream dout = new DataOutputStream(s.getOutputStream());

int r;

while((r=fin.read())!=-1){

dout.write(r);

}

System.out.println("\nFile transfer completed");

s.close();

ss.close();

}}
```

CLIENT

```
import java.io.*;

import java.net.*;

class Filereceiver

{

public static void main(String args[]) throws IOException{

Socket s=new Socket("127.0.0.1",7777);

if(s.isConnected())

{

System.out.println("This program is done by HARICHSELVAM 211191101045");

System.out.println("connected to server");

}

FileOutputStream fout= new FileOutputStream("Receive.txt"); DataInputStream din=new

DataInputStream(s.getInputStream());

int r;

while((r=din.read())!=-1)

{

fout.write((char)r);

}

s.close();

}

}
```

OUTPUT:



RESULT:

Thus, the java program for transferring file from one machine to another machine using TCP is executed and the output is verified successfully

Ex.No:5 **Date:**

CREATE A SOCKET (TCP) BETWEEN TWO COMPUTERS AND ENABLE FILE TRANSFER BETWEEN THEM

AIM:

To implement Remote Procedure Call (RPC) in Server-Client Model using Java

ALGORITHM:

SERVER

Step 1: Start the program.

Step 2: Create an unnamed socket for the server.

Step 3: Name the socket using bind () system call with the parameters server_sockfd and the server address (sin_addr and sin_sport).

Step 4: The server gets the method name from the client.

Step 5: Prints the output of the method.

Step 6: Stop the program execution.

CLIENT

Step 1: Start the program.

Step 2: Create an unnamed socket for client using socket

Step 3: Input the method name to be called from the server.

Step 4: Pass the data to the server.

Step 5: Stop the program execution.

PROGRAM:

SERVER

```
import java.io.*;

import java.net.*;

class Server {

public static void main(String[] args) throws Exception {

ServerSocket sersock = new ServerSocket(3000);

System.out.println("This program is done by HARICHSELVAM 211191101045");

System.out.println("Server ready");

Socket sock = sersock.accept();

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

OutputStream ostream = sock.getOutputStream();

PrintWriter pwrite = new PrintWriter(ostream, true);

InputStream istream = sock.getInputStream();

BufferedReader receiveRead = new BufferedReader(new InputStreamReader(istream));

String receiveMessage, sendMessage, fun;

int a, b, c;

while (true) {

fun = receiveRead.readLine();if (fun != null) {

System.out.println("Operation : " + fun);

}

a = Integer.parseInt(receiveRead.readLine());
```

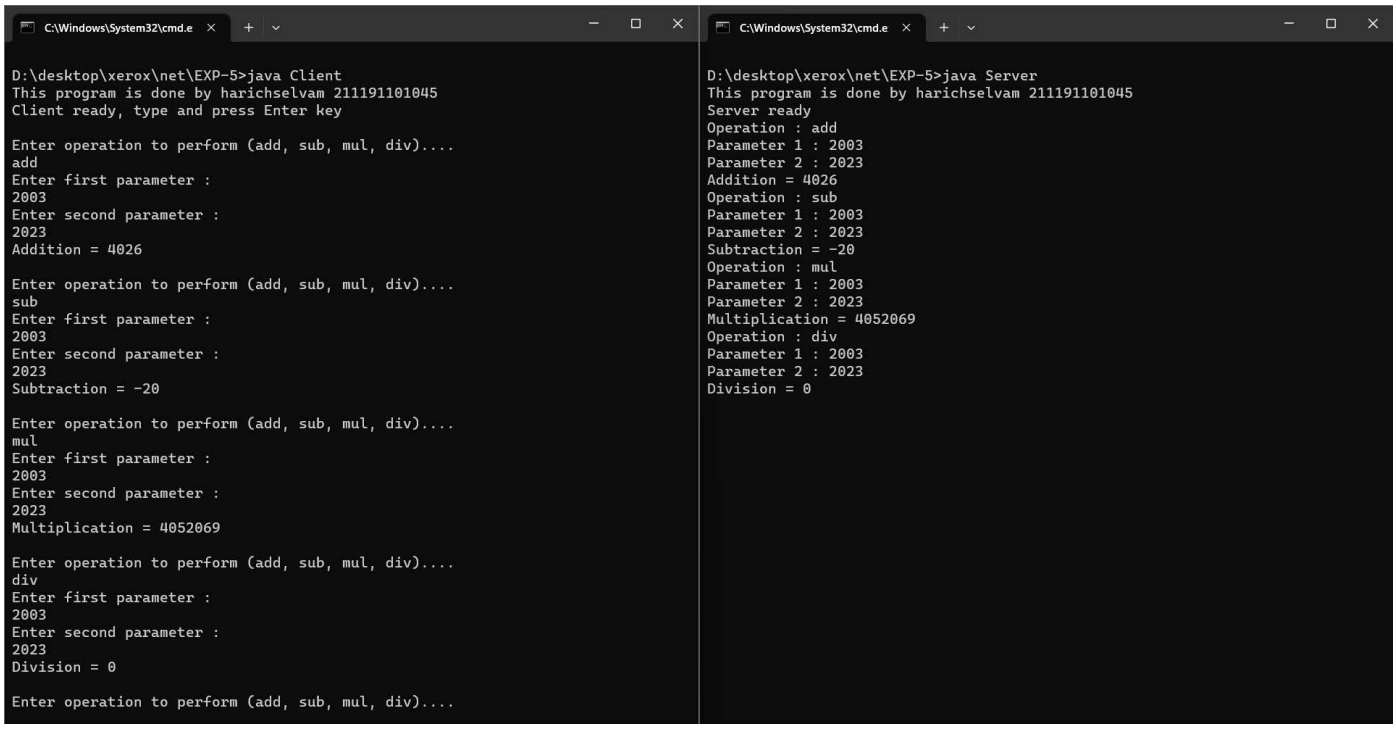
```
System.out.println("Parameter 1 : " + a);

b = Integer.parseInt(receiveRead.readLine());

System.out.println("Parameter 2 : " + b);

if (fun.compareTo("add") == 0) {

c = a + b;

System.out.println("Addition = " + c);pwrite.println("Addition = " + c);

}if (fun.compareTo("sub") == 0) {

c = a - b;

System.out.println("Substraction = " + c);

pwrite.println("Substraction = " + c);

}

if (fun.compareTo("mul") == 0) {

c = a * b;

System.out.println("Multiplication = " + c);

pwrite.println("Multiplication = " + c);

}

if (fun.compareTo("div") == 0) {

c = a / b;

System.out.println("Division = " + c);pwrite.println("Division = " + c);

}

System.out.flush();

}}}
```

CLIENT

```
import java.io.*;

import java.net.*;

class Client {

public static void main(String[] args) throws Exception {

Socket sock = new Socket("127.0.0.1", 3000);

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

OutputStream ostream = sock.getOutputStream();

PrintWriter pwrite = new PrintWriter(ostream, true);

InputStream istream = sock.getInputStream();

BufferedReader receiveRead = new BufferedReader(new InputStreamReader(istream));

System.out.println("This program is done by HARICHSELVAM 211191101045");

System.out.println("Client ready, type and press Enter key");

String receiveMessage, sendMessage, temp;

while (true) {

System.out.println("\nEnter operation to perform(add,sub,mul,div)....");

temp = keyRead.readLine();

sendMessage = temp.toLowerCase();

pwrite.println(sendMessage);

System.out.println("Enter first parameter :");

sendMessage = keyRead.readLine();

pwrite.println(sendMessage);
```

```
System.out.println("Enter second parameter : ");

sendMessage = keyRead.readLine();

pwrite.println(sendMessage);

System.out.flush();

if ((receiveMessage = receiveRead.readLine()) != null)

System.out.println(receiveMessage);

}

}

}
```

OUTPUT:



RESULT:

Thus, the Java program for implementing Remote Procedure Call is executed and the output is verified successfully

Ex.No:6 **Date:**

IMPLEMENTATION OF ARP/RARP

AIM:

To write a java program for simulating ARP and RARP protocols using TCP

ALGORITHM:

CLIENT

1.Start the program

2. Using socket, connection is established between client and server.

3.Get the IP address to be converted into MAC address.

4. Send this IP address to server.

5. Server returns the MAC address to client.

Server

1.Start the program

2. Accept the socket which is created by the client.

3.Server maintains the table in which IP and corresponding MAC addresses are stored.

4. Read the IP address which is send by the client.

5. Map the IP address with its MAC address and return the MAC address to client.

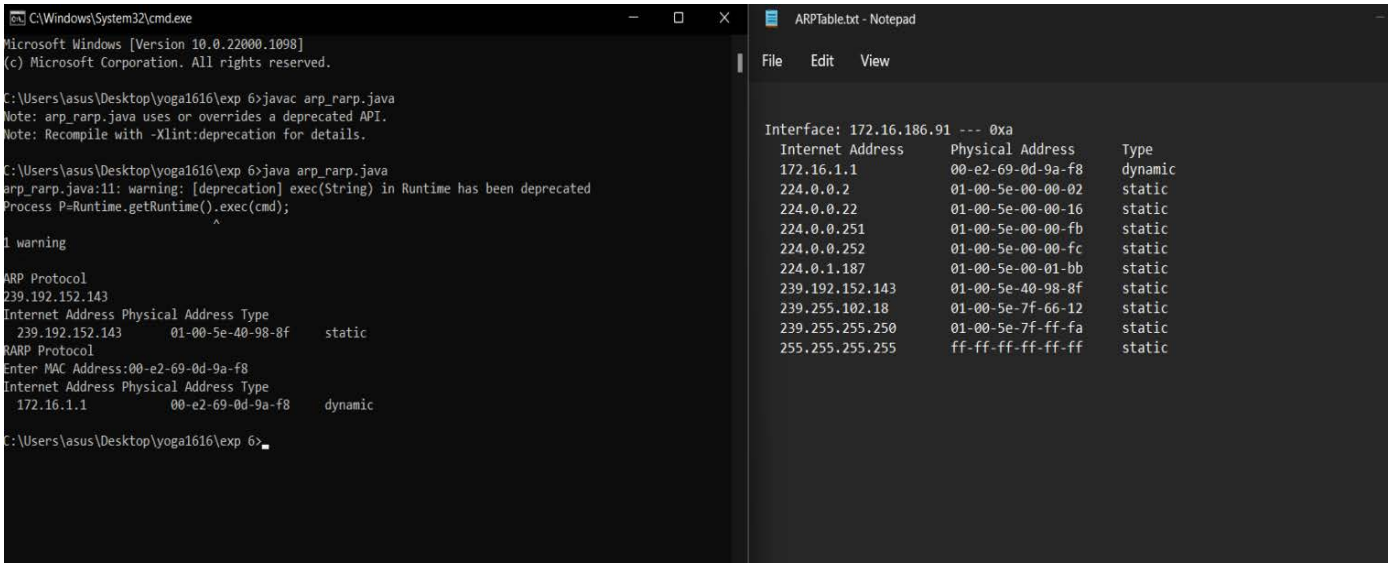
PROGRAM

```
import java.io.*;
import java.util.*;
public class arp_rarp
{
private static final String Command="arp -a";
public static void getARPTable(String cmd) throws Exception
{
File fp=new File("ARPTable.txt");
FileWriter fw=new FileWriter(fp);
BufferedWriter bw=new BufferedWriter(fw);
Process P=Runtime.getRuntime().exec(cmd);
Scanner S=new Scanner(P.getInputStream()).useDelimiter("\\A");
while(S.hasNext())
bw.write(S.next());
bw.close();fw.close();
}
public static void findMAC(String ip) throws Exception{
File fp=new File("ARPTable.txt");
FileReader fr=new FileReader(fp);
BufferedReader br=new BufferedReader(fr);
String line;
```

```
while((line=br.readLine())!=null)
{
if(line.contains(ip))
{
System.out.println("Internet Address Physical Address Type");
System.out.println(line);
break;
}}
if(line==null)
System.out.println("Not found");
fr.close();br.close();
}
public static void findIP(String mac) throws Exception
{
File fp=new File("ARPTable.txt");
FileReader fr=new FileReader(fp);
BufferedReader br=new BufferedReader(fr);
String line;
while((line=br.readLine())!=null){
if(line.contains(mac)){
System.out.println("Internet Address Physical Address Type");
System.out.println(line);
```

```
break;
}}
if(line==null)
System.out.println("Not Found");
fr.close();
br.close();
}
public static void main(String args[]) throws Exception{
System.out.println("This program is done by HARICHSELVAM 211191101045");
getARPTable(Command);
Scanner S=new Scanner(System.in);
System.out.println("ARP Protocol");
String IP=S.nextLine();
findMAC(IP);
System.out.println("RARP Protocol");
System.out.print("Enter MAC Address:");
String MAC=S.nextLine();
findIP(MAC);
}
}
```

OUTPUT:



RESULT:

Thus, the Java program for implementing ARP/RARP is executed and the output is verified successfully.

Ex.No:7 **Date:**

HTTP SOCKET PROGRAM TO DOWNLOAD A WEB PAGE.

AIM:

To write a HTTP socket program to download a web page.

ALGORITHM:

Step 1: Start the program

Step 2: Create a SocketHTTPClient.java file and add the main method

Step 3: Input the hostname and provide the default port number as 80

Step 4: Create a Socket by passing hostname and port number as parameters.

Step 4: Use the Print Writer class to retrieve the HTML content

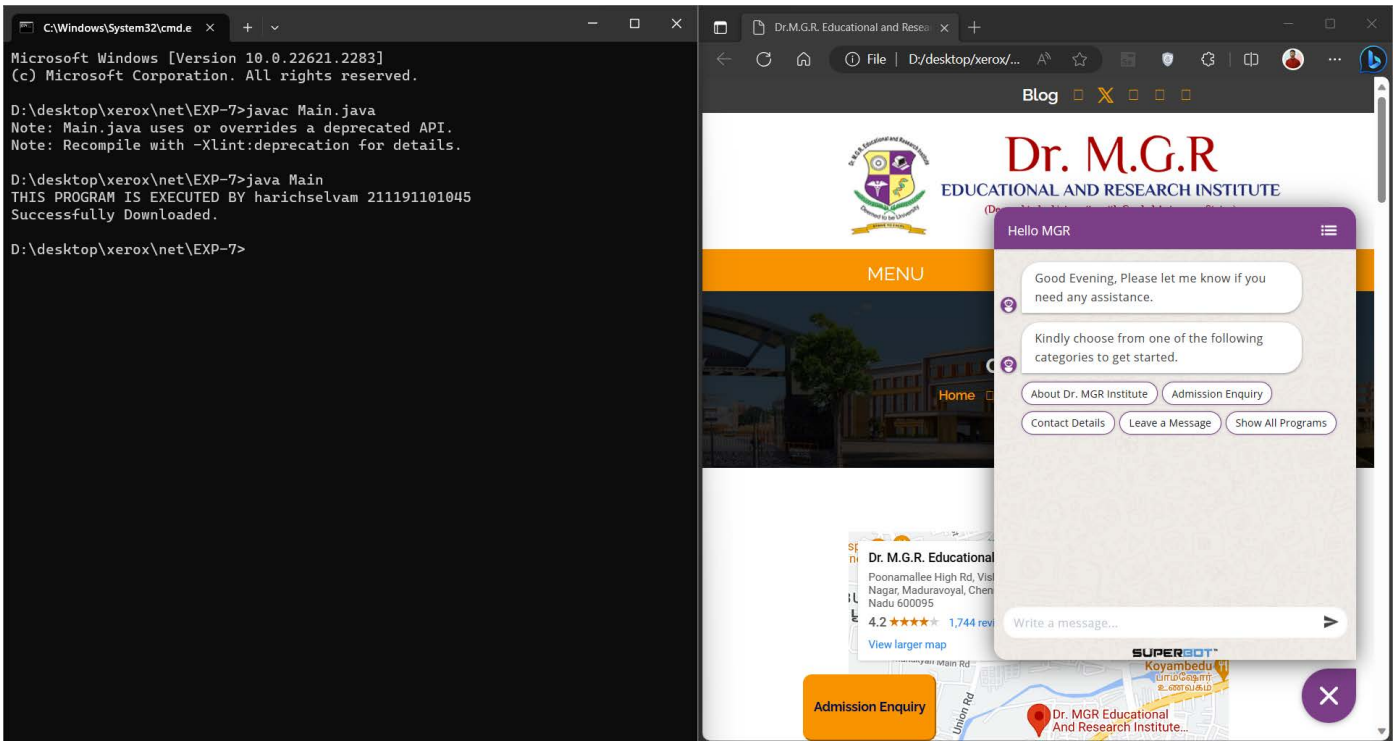
Step 5: Display the content in the console.

PROGRAM:

```
package normal;

import java.io.*;

import java.net.URL;

import java.net.MalformedURLException;


public class Main {

public static void DownloadWebPage(String webpage)

{

try {

URL url = new URL(webpage);

BufferedReader readr = new BufferedReader(new InputStreamReader(url.openStream()));

BufferedWriter writer = new BufferedWriter(new FileWriter("Data.html"));

String line;

while ((line = readr.readLine()) != null)

{

writer.write(line);

}

readr.close();

writer.close();

System.out.println("THIS PROGRAM IS EXECUTED BY  HARICHSELVAM 211191101045");

System.out.println("Successfully Downloaded.");
```

```
}
catch (MalformedURLException mue)
{
System.out.println("Malformed URL Exception raised");
}
catch (IOException ie)
{
System.out.println("IOException raised");
}
}
public static void main(String args[])
throws IOException
{
String url ="https://www.drmgrdu.ac.in/contact.php";
DownloadWebPage(url);
}
}
```

OUTPUT:



RESULT:

Thus, the Java program for downloading a web page using HTTP Socket is executed and the output is verified successfully.

Ex.No:8a **Date:**

USING RS232C

AIM:

To write a java program for file transfer client-server architecture using tcp-ip protocols.

ALGORITHM:

1. Import java files.
2. To create a socket in client and server.
3. The client established a connection through the given port to a server.
4. The client accepts the connection from a server.
5. The client communicates with the server to transfer the files.
6. Exit() command is used to stop the communication for file transfer.

PROGRAM:

```
import java.io.*;

import javax.comm.*;

public class Rs232c {

pulblic static void main( String arg[] ) {

try {

CommPortIdentifier ports = CommPortIdentifier.getPortIdentifier( "COM1" );

SerialPort port = ( SerialPort )ports.open( "RS232C", 1000 );

port.setSerialPortParams( 9600, SerialPort.DATABITS_8, SerialPort.STOPBITS_1,

SerialPort.PARITY_NONE );

port.setFlowControlMode( SerialPort.FLOWCONTROL_NONE );

OutputStream out = port.getOutputStream();

String msg = "Transferred";

out.write( msg.getBytes() );

out.flush();

out.close();

port.close();

}

catch( Exception e ) { System.out.println( "Error:" + e.getMessage() ); }

}

}
```

OUTPUT:

```
yoga@1616-LAPTOP MINGW64 ~/Desktop/Network/Lab exp 8
$ java Rs232c.java
Transfered

yoga@1616-LAPTOP MINGW64 ~/Desktop/Network/Lab exp 8
$
```

RESULT:

Thus, the java program using RS232C has been verified and executed successfully.

Ex.No:8b **Date:**

USING TCP\IP

AIM:

To write a java program for file transfer client-server architecture using tcp-ip protocols.

ALGORITHM:

1. Import java files

2. To create a socket in client and server.

3. The client established a connection through the given port to a server.

4. The client accepts the connection from a server.

5. The client communicates with the server to transfer the files.

6. Exit () command is used to stop the communication or file transfer.

PROGRAM:

Phonedirectory:

```
import java.util.*;
import java.io.*;
public class PhoneDirectory {
private Map pbMap = new HashMap();
public PhoneDirectory(String fileName) {
   // Initial content of the phone book
   // is read from the file
   // storing lines of the form: name phone_number
   try {
     BufferedReader br = new BufferedReader(
     new FileReader(fileName)); String line;
     while ((line = br.readLine()) != null) {
       String[] info = line.split(" +", 2);
       pbMap.put(info[0], info[1]);
     }} catch (Exception exc) {
       exc.printStackTrace();
       System.exit(1);
   }
 }
 // Returns the phone number of the given person
```

```
  public String getPhoneNumber(String name) {
    return (String) pbMap.get(name);
  }
  // Adds a new entry to the book
  // The result:
  // - true - the entry was added
  // - false - the entry was not added
  //           (the book associates another number with the given person)
  public boolean addPhoneNumber(String name, String num) {
    if (pbMap.containsKey(name)) return false;
    pbMap.put(name, num);
    return true;
  // Replaces the phone number of the given person
  // The result:
  // - true (success)
  // - false (failure - the given person does not exist)
  public boolean replacePhoneNumber(String name, String num) {
    if (!pbMap.containsKey(name)) return false;
    pbMap.put(name, num);
    return true;
  }
}
```

PhoneBookServer:

```
import java.net.*;
import java.io.*;
import java.util.regex.*;
public class PhoneBookServer  {
private PhoneDirectory pd = null;  // the map associating names and numbers
private ServerSocket ss = null;
private BufferedReader in = null;  // socket streams
private PrintWriter out = null;    // for communicating with the client
  public PhoneBookServer(PhoneDirectory pd, ServerSocket ss) {
    this.pd = pd;
this.ss = ss;
System.out.println("This program is done by HARICHSELVAM 211191101045");
System.out.println("Server started");
    System.out.println("on port: " + ss.getLocalPort());
    System.out.println("bind address: " + ss.getInetAddress());
    serviceConnections();  // listen to connections
  }
  // The method listens for connections made by clients.
  // After accepting a connection it creates a new Socket
  // and calls the method serviceRequest to handle the request
  private void serviceConnections() {
```

```
  boolean serverRunning = true;   // the server is running continously
  while (serverRunning) {
   try {
     Socket conn = ss.accept();  // listens and accepts connections
     System.out.println("Connection established");
     serviceRequests(conn);      // handle the request
   } catch (Exception exc)
  {
       exc.printStackTrace();
   }
  }                          // close the cosket
  try { ss.close(); } catch (Exception exc) {}
}
// the pattern for parsing requests
private static Pattern reqPatt = Pattern.compile(" +", 3);
// Server messages.
// The corresponding codes are indices of the array.
private static String msg[] = { "Ok", "Invalid request", "Not found",
                                "Couldn't add - entry already exists",
                                "Couldn't replace non-existing entry",
                               };
// Handling client's request
```

```
private void serviceRequests(Socket connection)
                        throws IOException {
  try {
    in = new BufferedReader(                  // create the streams
            new InputStreamReader(
               connection.getInputStream()));
    out = new PrintWriter(
             connection.getOutputStream(), true);
    // Parse the request
    for (String line; (line = in.readLine()) != null; ) {
      String resp;                          // answer
      String[] req = reqPatt.split(line, 3);
      String cmd = req[0];                  // command
      if (cmd.equals("bye")) {        // end of message
        writeResp(0, null);
        break;
      }
      else if (cmd.equals("get")) {   // get the number
        if (req.length != 2) writeResp(1, null);
        else {
          String phNum = (String) pd.getPhoneNumber(req[1]);
          if (phNum == null) writeResp(2, null);
```

```
        else writeResp(0, phNum);
      }
    }
    else if (cmd.equals("add"))  {  // add a number
      if (req.length != 3) writeResp(1, null);
      else {
        boolean added = pd.addPhoneNumber(req[1], req[2]);
        if (added) writeResp(0, null);
        else writeResp(3, null);
      }
    }
    else if (cmd.equals("replace"))  {  // change the number
      if (req.length != 3) writeResp(1, null);
      else {
        boolean replaced = pd.replacePhoneNumber(req[1], req[2]);
        if (replaced) writeResp(0, null);
        else writeResp(4, null);
      }
    }
    else writeResp(1, null);            // invalid request
  }
}
```

```
catch (Exception exc)
{
        exc.printStackTrace();
    } finally
{
        try {                       // close the streams
          in.close();                 // and the socket
          out.close();
          connection.close();
          connection = null;
        } catch (Exception exc) { }
    }
  }
  // Pass the response to the client
  private void writeResp(int rc, String addMsg) throws IOException {
    out.println(rc + " " + msg[rc]);
    if (addMsg != null) out.println(addMsg);
  }

  public static void main(String[] args) {
    PhoneDirectory pd = null;
    ServerSocket ss = null;
```

```
    try {
      String phdFileName = args[0];
      String host = args[1];
      int port = Integer.parseInt(args[2]);
      pd = new PhoneDirectory(phdFileName); // create the map (read data from the file)
      InetSocketAddress isa = new InetSocketAddress(host, port);
      ss =  new ServerSocket();            // create the server socket
      ss.bind(isa);                        // and bind it to the address
    } catch(Exception exc) {
        exc.printStackTrace();
        System.exit(1);
    }
    new PhoneBookServer(pd, ss);
  }
}
```

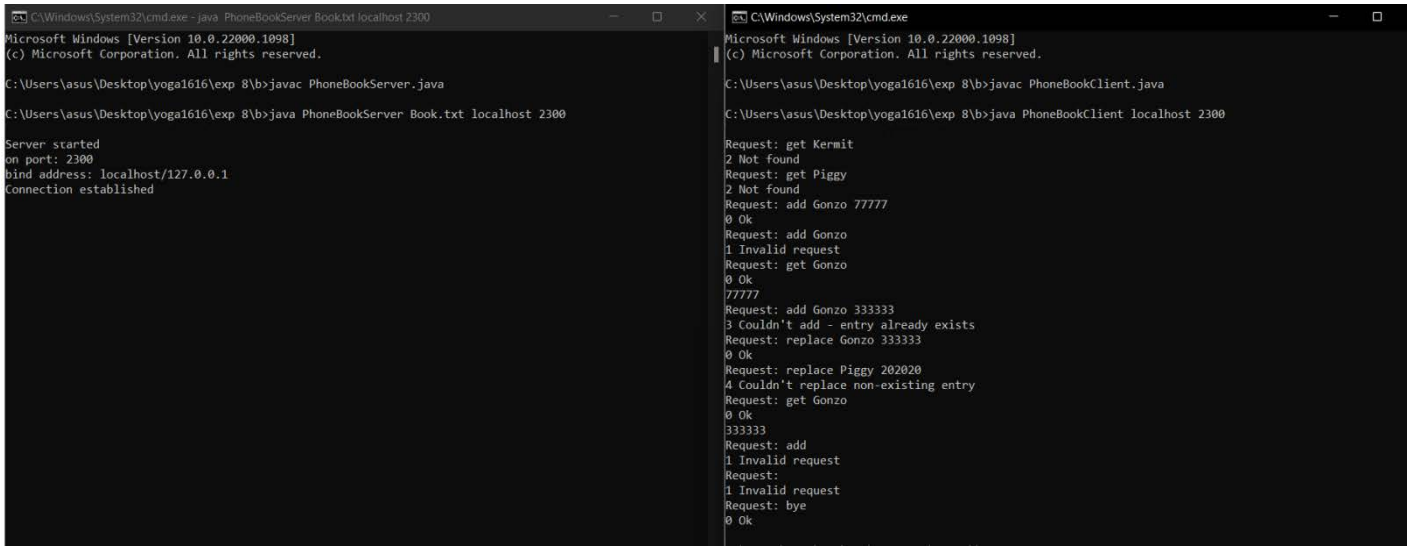
PhoneBookClient:

```
import java.net.*;

import java.io.*;

public class PhoneBookClient

{

  private Socket sock = null;

  private PrintWriter out = null;

  private BufferedReader in = null;

  public PhoneBookClient(String host, int port) {

    try {

      sock = new Socket(host, port);

      out = new PrintWriter(sock.getOutputStream(), true);

      in = new BufferedReader(

              new InputStreamReader(

                 sock.getInputStream()));

           System.out.println("This Program is done by HARICHSELVAM  211191101045");

      makeRequest("get Kermit");

                 makeRequest("get Piggy");

                 makeRequest("add Gonzo 77777");

                 makeRequest("add Gonzo");

                 makeRequest("get Gonzo");

                 makeRequest("add Gonzo 333333");
```

```
        makeRequest("replace Gonzo 333333");
        makeRequest("replace Piggy 202020");
        makeRequest("get Gonzo");
        makeRequest("add");
        makeRequest("");
        makeRequest("bye");
        in.close();
        out.close();
        sock.close();
      } catch (UnknownHostException e)
{
          System.err.println("Unknown host: "+host);
          System.exit(2);
      } catch (IOException e) {
          System.err.println("I/O error for");
          System.exit(3);
      }
   catch (Exception exc) {
          exc.printStackTrace();
          System.exit(4);
      }
    }
```

```
  private boolean makeRequest(String req) throws IOException {
    System.out.println("Request: " + req);
    out.println(req);
    String resp = in.readLine();
    System.out.println(resp);
    boolean ok = resp.startsWith("0");
    if (req.startsWith("get") && ok)
       System.out.println(in.readLine());
    return ok;
  }

  public static void main(String[] args) {
    new PhoneBookClient(args[0], Integer.parseInt(args[1]));
  }
}
```

OUTPUT:



RESULT:

Thus, the java program using TCP\IP has been verified and executed successfully.

Ex.No:9 **Date:**

TO IMPLEMENT RMI (REMOTE METHOD INVOCATIO)

AIM:

To implement the RMI using the given protocol.

ALGORITHM:

1. Start the program by operating the socket.
2. It initiates a connection with remove Virtual Machine (JVM).
3. It writes and transmits(marshals) the parameters to the remote Virtual Machine (JVM).
4. It waits (unmarshal)the return value or exception.
5. It finally,returns the value to the caller.

PROGRAM:

SERVER:

```
import java.rmi.*;
import java.rmi.server.*;
 class RMIServer extends UnicastRemoteObject implements myinterface
{
public RMIServer()throws RemoteException{
System.out.println("Remote Server is running Now.!!");
}
public static void main(String arg[]){
System.out.println("This program is done by HARICHSELVAM 211191101045"); try{
RMIServer p=new RMIServer();Naming.rebind("rmiInterface",p);
}catch(Exception e)
{ System.out.println("Exception occurred : "+e.getMessage());
 }}
public String countInput(String input) throws RemoteExceptio{
System.out.println("Received your input "+ input+"  at server!!");
String reply;
reply="You have typed "+ input.length() +"  letters!!";return reply;
}
}
```

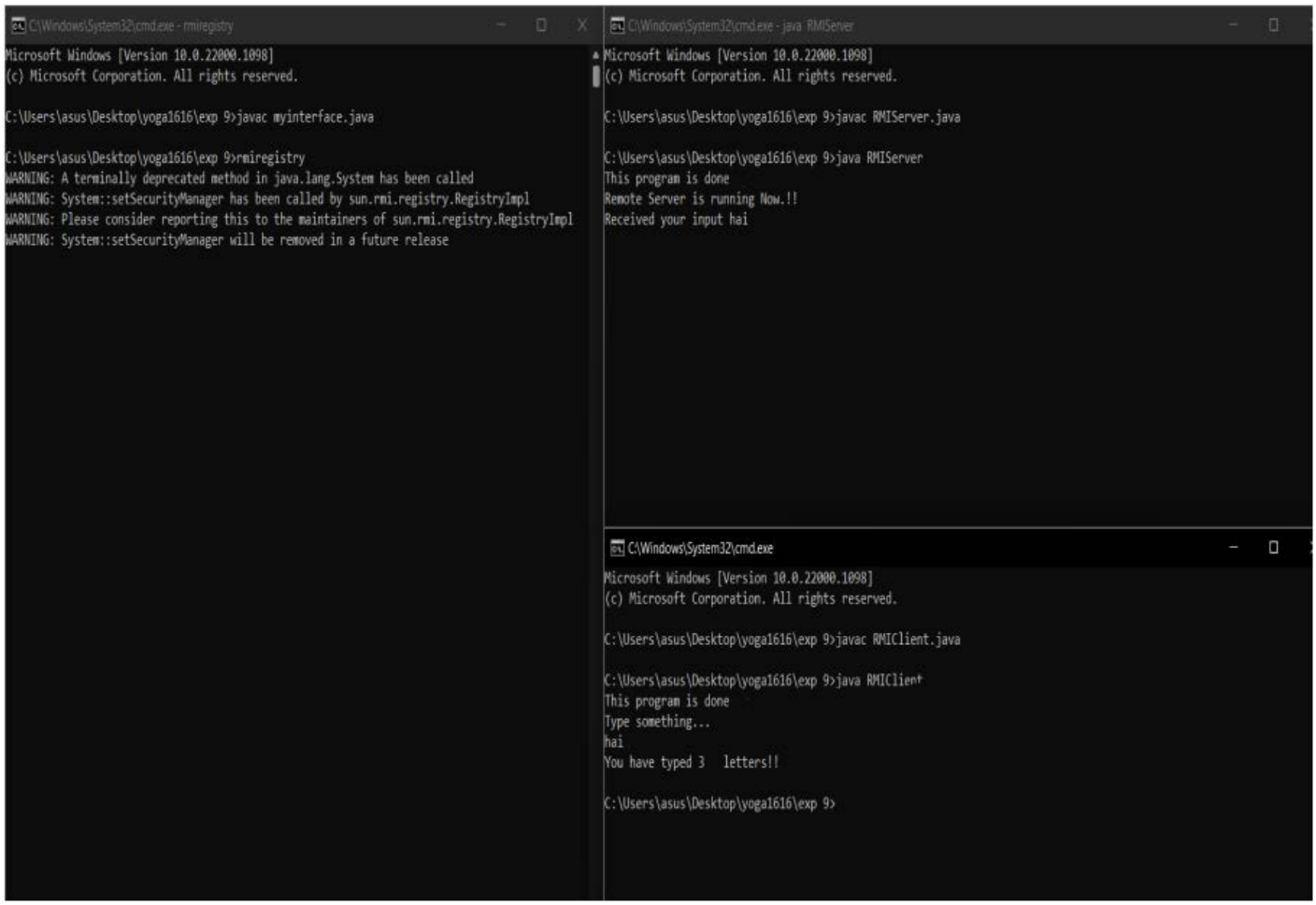
CLIENT:

```
import java.rmi.*;
import java.io.*;
public class RMIClient
{
public static void  main(String args[])
{
System.out.println("This program is done by HARICHSELVAM 211191101045"); try
{
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
myinterface p=( myinterface)Naming.lookup("rmiInterface");
System.out.println("Type something...");
String input=br.readLine();
System.out.println(p.countInput(input));
}
catch(Exception e)
{
System.out.println("Exception occurred : "+e.getMessage());


}
}
}
```

MY INTERFACE

```
import java.rmi.*;

public interface myinterface extends Remote

{

public String countInput(String input)throws RemoteException;

}
```

OUTPUT:



RESULT:

Thus, the Java program for implementing Remote Method Invocation is executed and the output is verified successfully.

Ex. No:10 **Date:**

DEMONSTRATION OF NETWORKS SIMULATORS

AIM:

To write a program for the demonstration of networks simulators.

ALGORITHM:

1. Start the program.
2. Import necessary packages.
3. Create the frame and define necessary parameters for frame.
4. Using socket get the local host.
5. Open the connection using IO Buffer stream.
6. Close the connection.

PROGRAM:

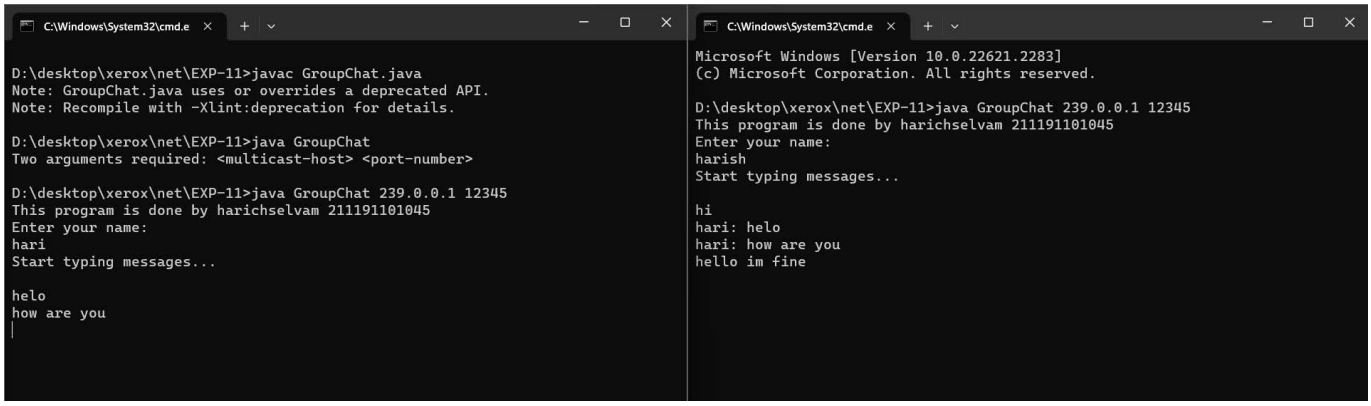
```
import java.net.*;

import java.io.*;

import java.util.*;

public class GroupChat

{

private static final String TERMINATE = "Exit";

static String name;

static volatile boolean finished = false;

public static void main(String[] args){

if (args.length != 2)

System.out.println("Two arguments required: <multicast-host> <port-number>");

else{

try{

InetAddress group = InetAddress.getByName(args[0]);

int port = Integer.parseInt(args[1]);

Scanner sc = new Scanner(System.in);

System.out.println("This program is done by HARICHSELVAM 211191101045");

System.out.println("Enter your name: ");

name = sc.nextLine();

MulticastSocket socket = new MulticastSocket(port);

socket.setTimeToLive(0);
```

```
socket.joinGroup(group);
Thread t = new Thread(new
ReadThread(socket,group,port));
t.start();
System.out.println("Start typing messages...\n");
while(true)
{
String message;
message = sc.nextLine();
if(message.equalsIgnoreCase(GroupChat.TERMINATE))
{
finished = true;
socket.leaveGroup(group);
socket.close();
break;
}
message = name + ": " + message;
byte[] buffer = message.getBytes();
DatagramPacket datagram = new
DatagramPacket(buffer,buffer.length,group,port);
socket.send(datagram);
}
```

```
catch(SocketException se)
{
System.out.println("Error creating socket");
se.printStackTrace();
}
catch(IOException ie{
System.out.println("Error reading/writing from/to socket");
ie.printStackTrace();
}}
}}
class ReadThread implements Runnable
{
private MulticastSocket socket;
private InetAddress group;private int port;
private static final int MAX_LEN = 1000;
ReadThread(MulticastSocket socket,InetAddress group,int port)
{
this.socket = socket;
this.group = group;this.port = port;
}
@Override
public void run()
```

```
{
while(!GroupChat.finished)
{
byte[] buffer = new byte[ReadThread.MAX_LEN];
DatagramPacket datagram = new
DatagramPacket(buffer,buffer.length,group,port);
String message;
try{
socket.receive(datagram);
message = new
String(buffer,0,datagram.getLength(),"UTF-8");
if(!message.startsWith(GroupChat.name))
System.out.println(message);
}
catch(IOException e)
{
System.out.println("Socket closed!");
}
}
}
}
```

OUTPUT:



RESULT:

Thus, the java program for implementing the demonstration of network simulators has been verified and executed successfully.