

Experiment 1

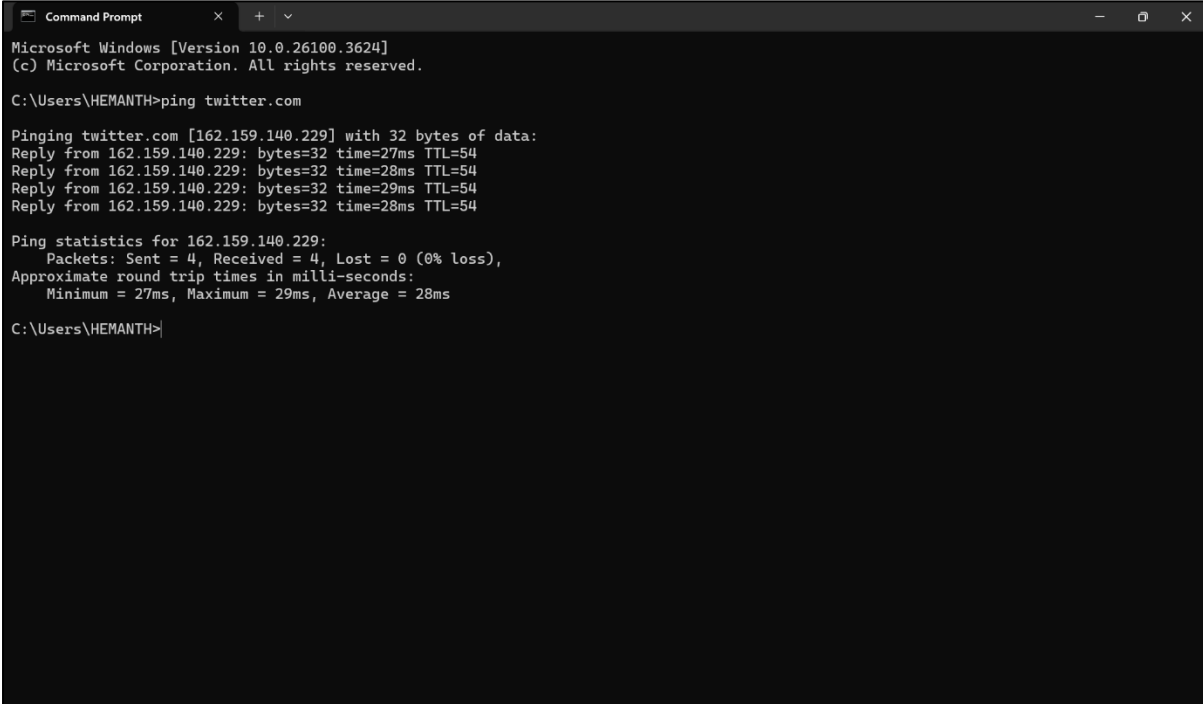
AIM: To study of basic network commands and network configuration commands

- a. Ping
- b. Tracent
- c. Ipconfig
- d. Hostname
- e. Nslookup
- f. netstat

ping: It is one of the basic networking commands to test the connection between the local machine and the host server. This command sends a small amount of data to the host server, and in return the host server sends a replay to the computer information like the ip address of the host server, the amount of data sent, time to live. And time needed for sending and the data are recorded and displayed to the user.

Ex:

C:\Users\HEMANTH>ping twitter.com



```
Command Prompt
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH>ping twitter.com

Pinging twitter.com [162.159.140.229] with 32 bytes of data:
Reply from 162.159.140.229: bytes=32 time=27ms TTL=54
Reply from 162.159.140.229: bytes=32 time=28ms TTL=54
Reply from 162.159.140.229: bytes=32 time=29ms TTL=54
Reply from 162.159.140.229: bytes=32 time=28ms TTL=54

Ping statistics for 162.159.140.229:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 27ms, Maximum = 29ms, Average = 28ms

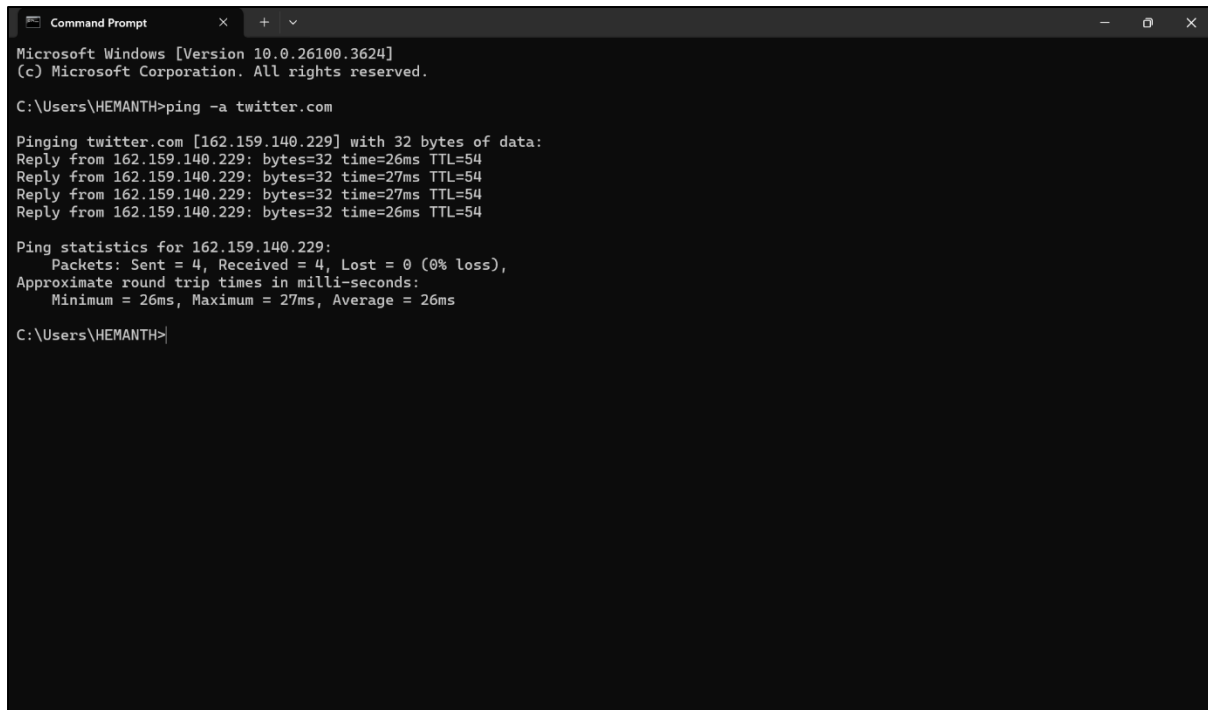
C:\Users\HEMANTH>
```

-a: The “-a” option resolves the hostname to the respective ip address

The command ping -a is used to perform a ping test while attempting the resolve the hostname of the target ip address.

Ex:

C:\Users\HEMANTH>ping -a twitter.com



```
Command Prompt
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH>ping -a twitter.com

Pinging twitter.com [162.159.140.229] with 32 bytes of data:
Reply from 162.159.140.229: bytes=32 time=26ms TTL=54
Reply from 162.159.140.229: bytes=32 time=27ms TTL=54
Reply from 162.159.140.229: bytes=32 time=27ms TTL=54
Reply from 162.159.140.229: bytes=32 time=26ms TTL=54

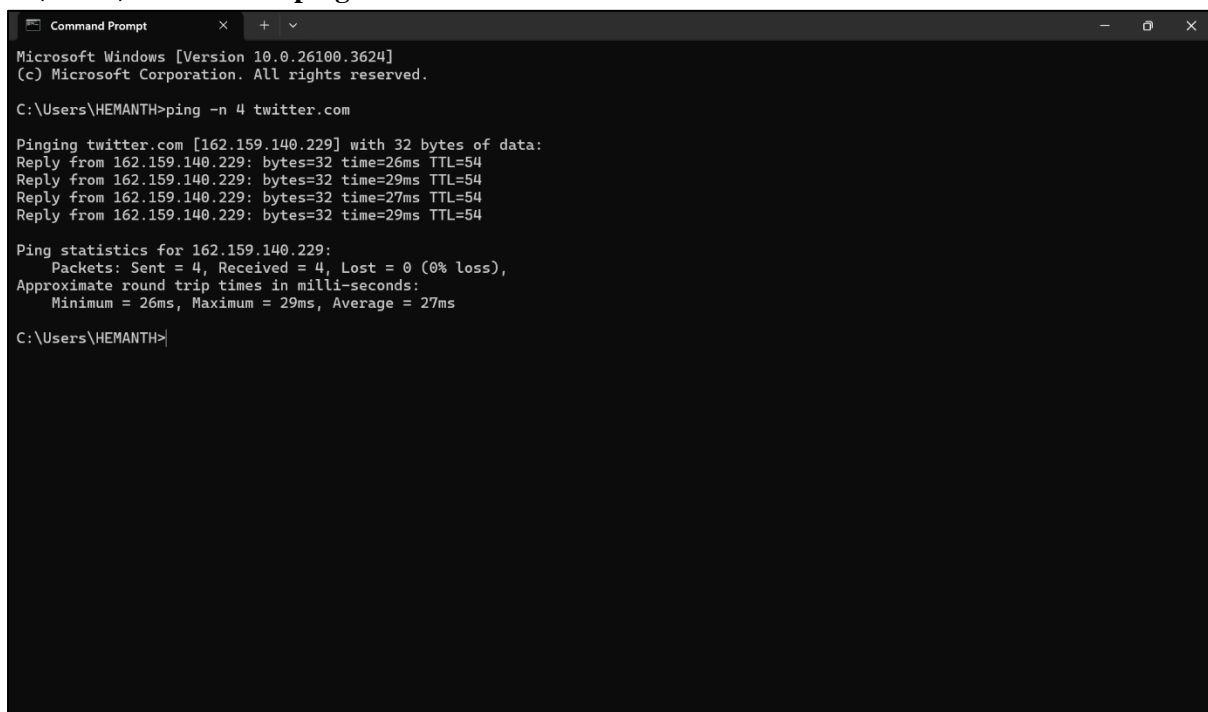
Ping statistics for 162.159.140.229:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 26ms, Maximum = 27ms, Average = 26ms

C:\Users\HEMANTH>
```

-n count: The -n <count> option in the ping command specifies the number of echo request packets to send. This is useful when you want to control the number of ping attempts instead of running indefinitely.

Ex:

C:\Users\HEMANTH>ping -n 4 twitter.com



```
Command Prompt
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH>ping -n 4 twitter.com

Pinging twitter.com [162.159.140.229] with 32 bytes of data:
Reply from 162.159.140.229: bytes=32 time=26ms TTL=54
Reply from 162.159.140.229: bytes=32 time=29ms TTL=54
Reply from 162.159.140.229: bytes=32 time=27ms TTL=54
Reply from 162.159.140.229: bytes=32 time=29ms TTL=54

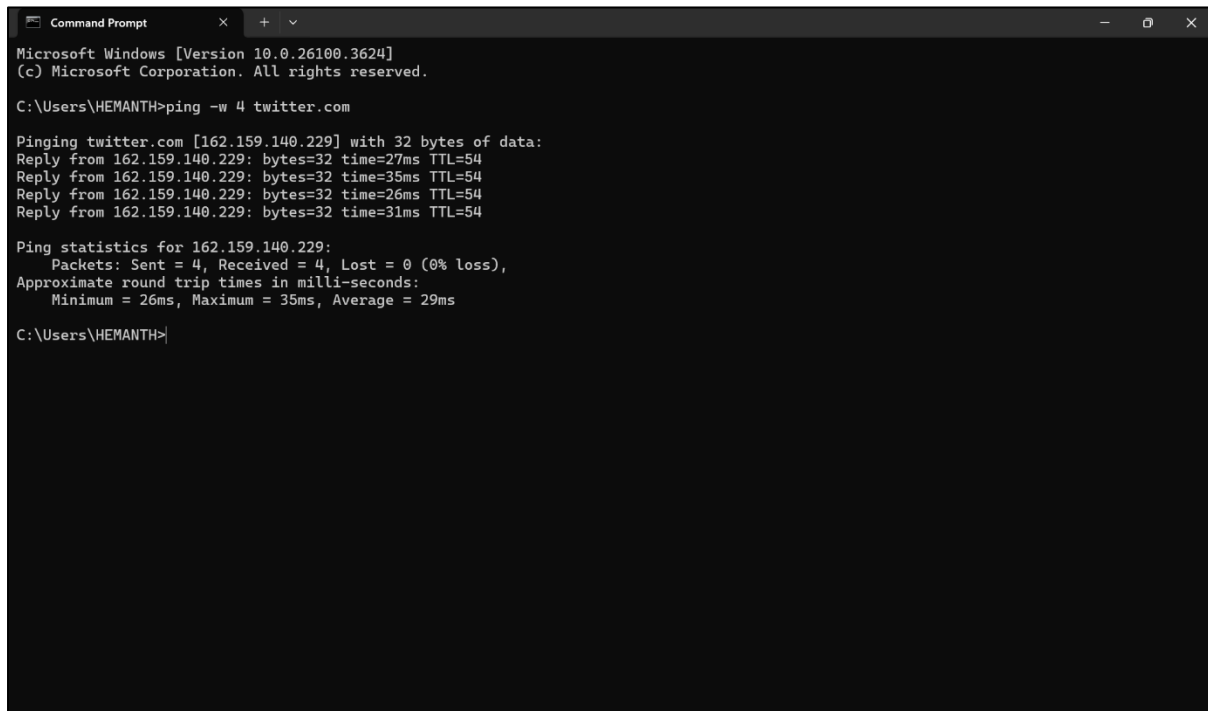
Ping statistics for 162.159.140.229:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 26ms, Maximum = 29ms, Average = 27ms

C:\Users\HEMANTH>
```

-w timeout: The -w <timeout> option in the ping command sets a timeout (in milliseconds) for each reply. If the response is not received within the specified time, the request is considered lost.

Ex:

C:\Users\HEMANTH>ping -w 4 twitter.com



```
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH>ping -w 4 twitter.com

Pinging twitter.com [162.159.140.229] with 32 bytes of data:
Reply from 162.159.140.229: bytes=32 time=27ms TTL=54
Reply from 162.159.140.229: bytes=32 time=35ms TTL=54
Reply from 162.159.140.229: bytes=32 time=26ms TTL=54
Reply from 162.159.140.229: bytes=32 time=31ms TTL=54

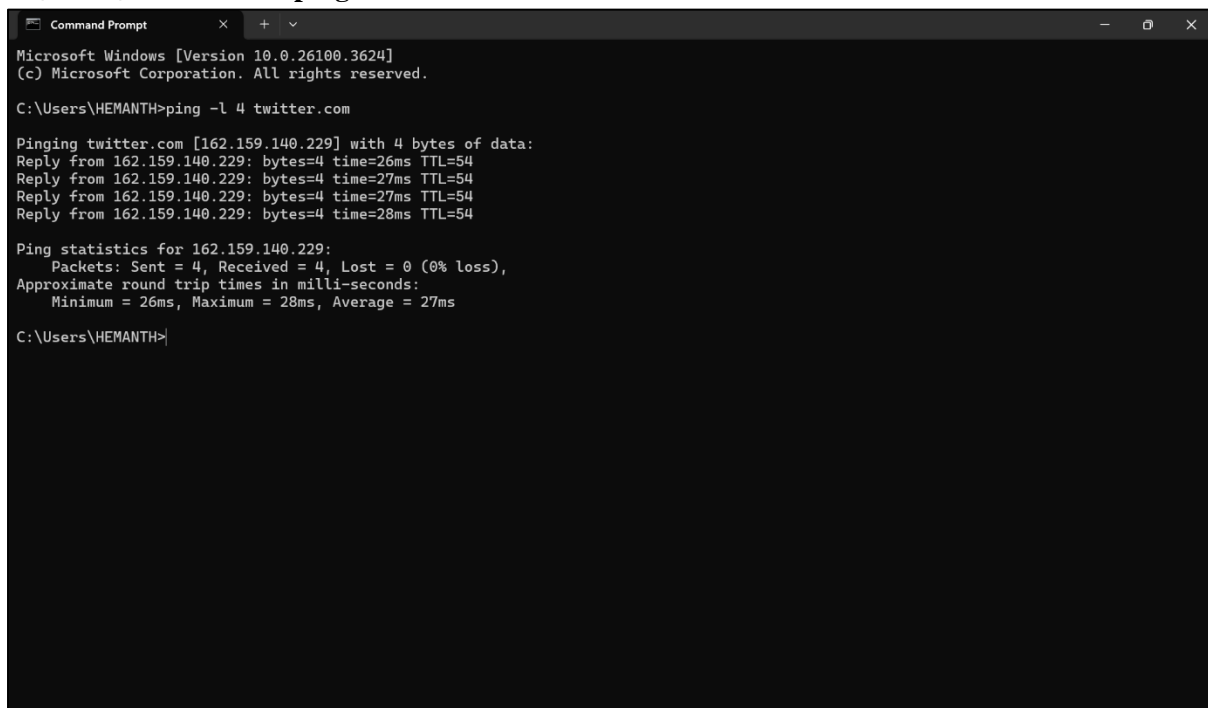
Ping statistics for 162.159.140.229:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 26ms, Maximum = 35ms, Average = 29ms

C:\Users\HEMANTH>
```

-l size: The **-l <size>** option in the ping command specifies the size of the packet (in bytes) to send. This is useful for testing network performance and identifying MTU (Maximum Transmission Unit) limits.

Ex:

C:\Users\HEMANTH>ping -l 4 twitter.com



```
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH>ping -l 4 twitter.com

Pinging twitter.com [162.159.140.229] with 4 bytes of data:
Reply from 162.159.140.229: bytes=4 time=26ms TTL=54
Reply from 162.159.140.229: bytes=4 time=27ms TTL=54
Reply from 162.159.140.229: bytes=4 time=27ms TTL=54
Reply from 162.159.140.229: bytes=4 time=28ms TTL=54

Ping statistics for 162.159.140.229:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 26ms, Maximum = 28ms, Average = 27ms

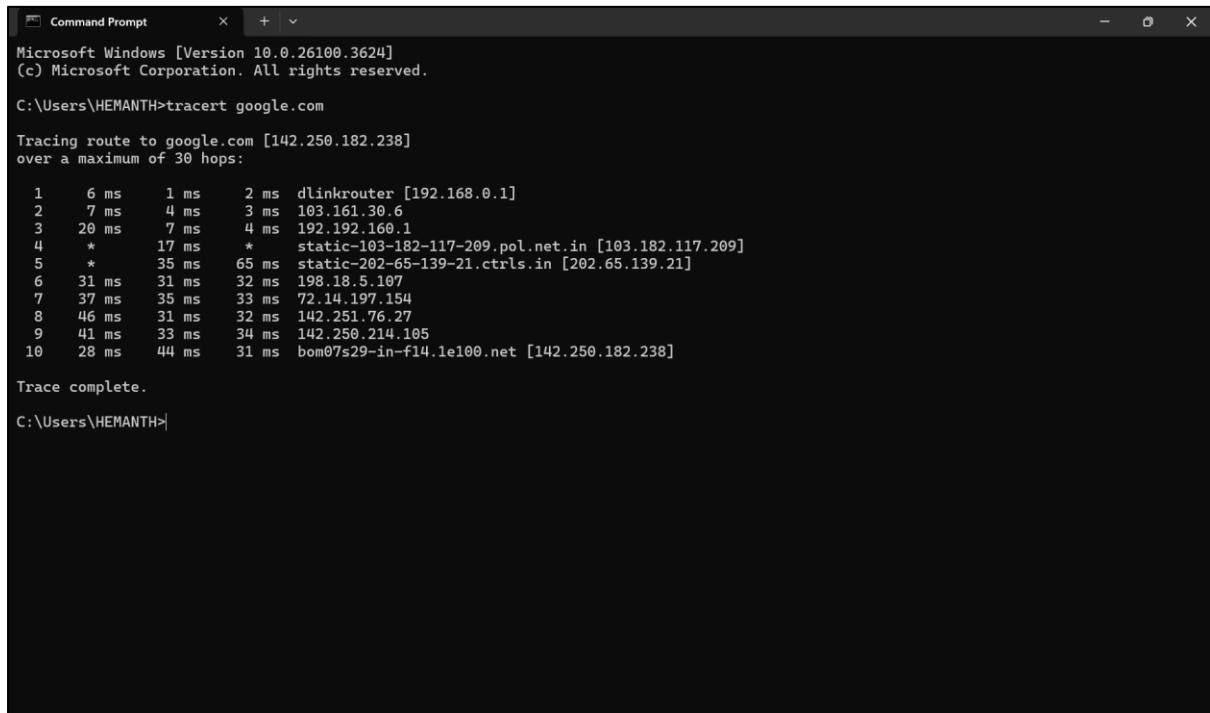
C:\Users\HEMANTH>
```

Tracert: The traceroute command (or tracert in Windows) is used to trace the route that packets take from the source to the destination across a network. It shows each intermediate hop (router or server)

the packets pass through on their way to the destination.

Ex:

C:\Users\HEMANTH>tracert google.com



```
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH>tracert google.com

Tracing route to google.com [142.250.182.238]
over a maximum of 30 hops:

  0  6 ms   1 ms   2 ms  dlinkrouter [192.168.0.1]
  1  7 ms   4 ms   3 ms  103.161.30.6
  2 20 ms   7 ms   4 ms  192.192.160.1
  3  *      17 ms  *      static-103-182-117-209.pol.net.in [103.182.117.209]
  4  *      35 ms  65 ms  static-202-65-139-21.ctrls.in [202.65.139.21]
  5 31 ms  31 ms  32 ms  198.18.5.107
  6 37 ms  35 ms  33 ms  72.14.197.154
  7 46 ms  31 ms  32 ms  142.251.76.27
  8 41 ms  33 ms  34 ms  142.250.214.105
  9 28 ms  44 ms  31 ms  bom07s29-in-f14.1e100.net [142.250.182.238]

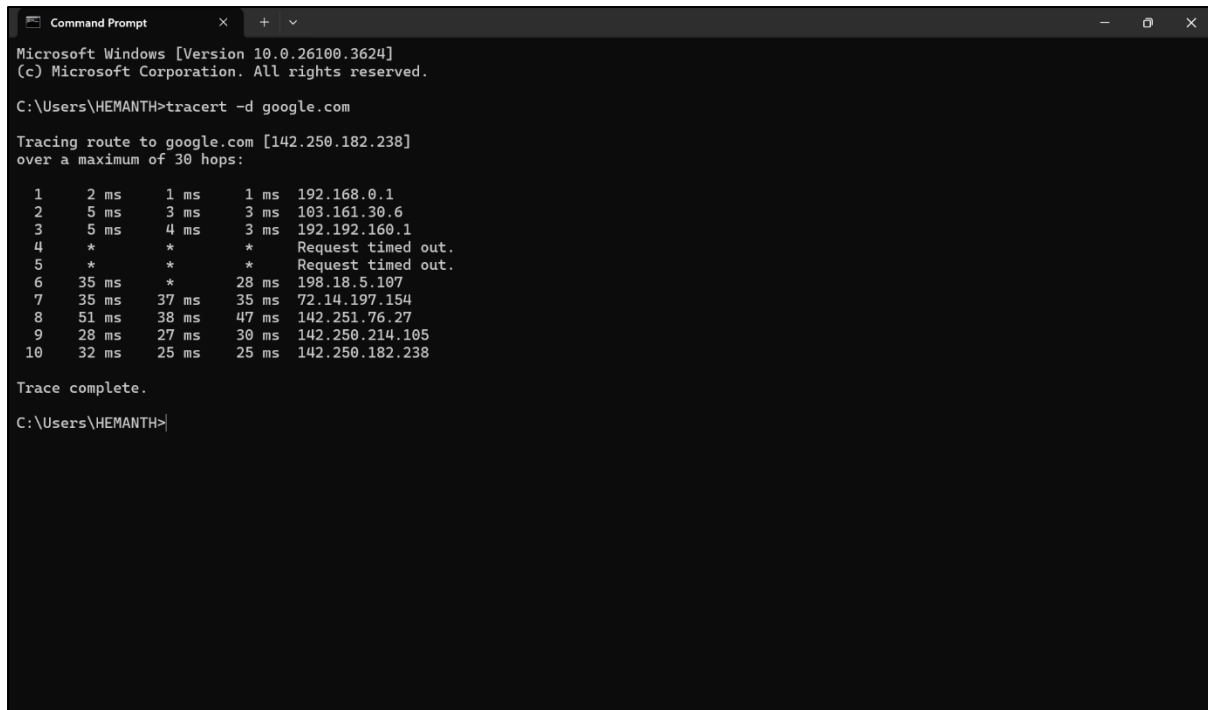
Trace complete.

C:\Users\HEMANTH>
```

-d: The -d option in the tracert command (Windows) is used to disable reverse DNS lookups during the trace process. By default, tracert attempts to resolve IP addresses to domain names (reverse DNS lookup) for each hop. Using the -d flag prevents this, which can speed up the trace process, as it avoids waiting for DNS queries to resolve.

Ex:

C:\Users\HEMANTH>tracert -d google.com



```

Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH>tracert -d google.com

Tracing route to google.com [142.250.182.238]
over a maximum of 30 hops:

  0  2 ms    1 ms    1 ms    192.168.0.1
  1  5 ms    3 ms    3 ms    103.161.30.6
  2  5 ms    4 ms    3 ms    192.192.160.1
  3  *        *        *        Request timed out.
  4  *        *        *        Request timed out.
  5  35 ms   *        28 ms   198.18.5.107
  6  35 ms   37 ms   35 ms   72.14.197.154
  7  51 ms   38 ms   47 ms   142.251.76.27
  8  28 ms   27 ms   30 ms   142.250.214.105
  9  32 ms   25 ms   25 ms   142.250.182.238

Trace complete.

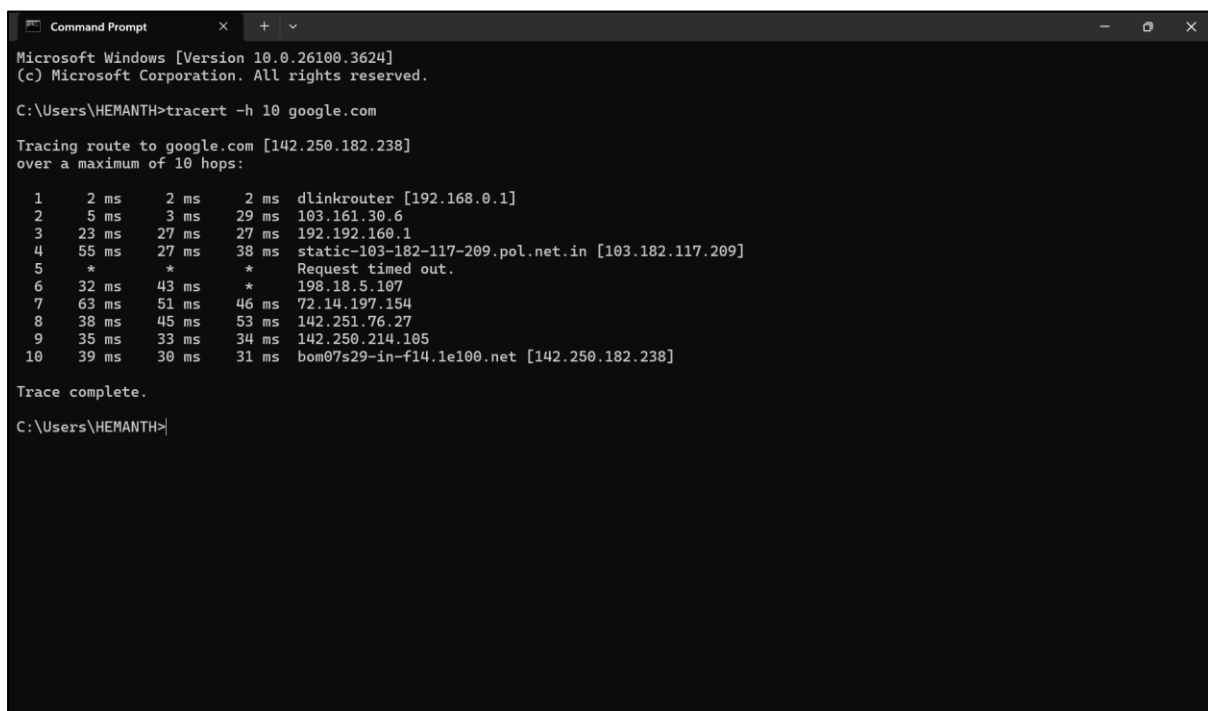
C:\Users\HEMANTH>

```

-h: The -h option in the tracert command (Windows) is used to specify the maximum number of hops that the traceroute should attempt before stopping. The default maximum hop count is 30, but you can change it with the -h flag.

Ex:

C:\Users\HEMANTH>tracert -h 10 google.com



```

Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH>tracert -h 10 google.com

Tracing route to google.com [142.250.182.238]
over a maximum of 10 hops:

  0  2 ms    2 ms    2 ms    dlinkrouter [192.168.0.1]
  1  5 ms    3 ms    29 ms   103.161.30.6
  2  23 ms   27 ms   27 ms   192.192.160.1
  3  55 ms   27 ms   38 ms   static-103-182-117-209.pol.net.in [103.182.117.209]
  4  *        *        *        Request timed out.
  5  32 ms   43 ms   *        198.18.5.107
  6  63 ms   51 ms   46 ms   72.14.197.154
  7  38 ms   45 ms   53 ms   142.251.76.27
  8  35 ms   33 ms   34 ms   142.250.214.105
  9  39 ms   30 ms   31 ms   bom07s29-in-f14.1e100.net [142.250.182.238]

Trace complete.

C:\Users\HEMANTH>

```

-w: The -w option in the tracert command (Windows) is used to set the timeout in milliseconds for each reply. This determines how long tracert should wait for a response from each hop before timing out.

Ex:

C:\Users\HEMANTH>tracert -w 2000 google.com

```

Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH>tracert -w 2000 google.com

Tracing route to google.com [142.250.182.238]
over a maximum of 30 hops:

  1  4 ms  1 ms  1 ms  dlinkrouter [192.168.0.1]
  2  5 ms  2 ms  2 ms  103.161.30.6
  3  12 ms  9 ms  4 ms  192.192.160.1
  4  *  15 ms  *  static-103-182-117-209.pol.net.in [103.182.117.209]
  5  *  *  *  Request timed out.
  6  29 ms  *  27 ms  198.18.5.107
  7  34 ms  32 ms  32 ms  72.14.197.154
  8  32 ms  29 ms  34 ms  142.251.76.27
  9  32 ms  26 ms  29 ms  142.250.214.105
 10  29 ms  25 ms  29 ms  bom07s29-in-f14.1e100.net [142.250.182.238]

Trace complete.

C:\Users\HEMANTH>

```

-4: The -4 option in the tracert command (Windows) forces the traceroute to use only IPv4 addresses, even if the destination supports both IPv4 and IPv6

Ex:

C:\Users\HEMANTH>tracert -4 google.com

```

Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH>tracert -4 google.com

Tracing route to google.com [142.250.192.46]
over a maximum of 30 hops:

  1  2 ms  2 ms  2 ms  dlinkrouter [192.168.0.1]
  2  5 ms  4 ms  3 ms  103.161.30.6
  3  6 ms  4 ms  3 ms  192.192.160.1
  4  12 ms  *  *  static-103-182-117-209.pol.net.in [103.182.117.209]
  5  35 ms  32 ms  27 ms  static-202-65-139-21.ctrls.in [202.65.139.21]
  6  30 ms  38 ms  25 ms  198.18.5.107
  7  43 ms  29 ms  30 ms  72.14.198.36
  8  32 ms  29 ms  27 ms  192.178.110.223
  9  39 ms  33 ms  32 ms  142.250.210.183
 10  33 ms  31 ms  35 ms  bom12s15-in-f14.1e100.net [142.250.192.46]

Trace complete.


C:\Users\HEMANTH>

```

Hostname: The hostname command is used to display or change the name of a computer on a network.

Ex:

C:\Users\HEMANTH>hostname



```
Command Prompt
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

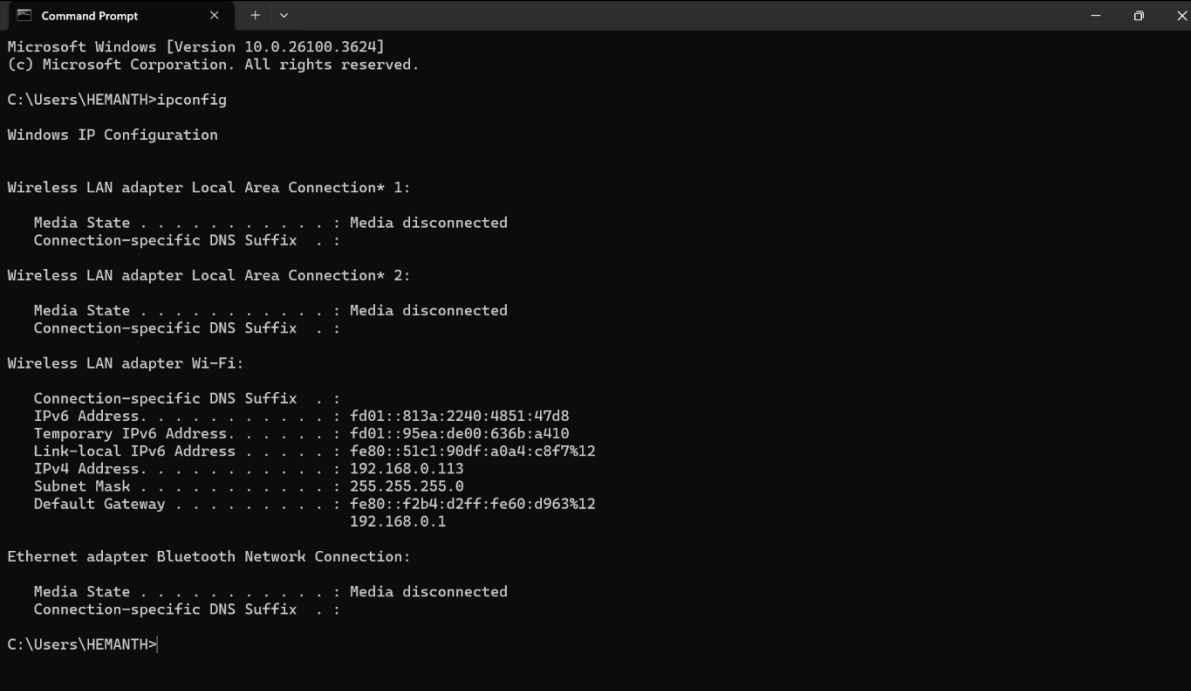
C:\Users\HEMANTH>hostname
LAPTOP-KJ3VQD1I

C:\Users\HEMANTH>
```

IPConfig: The ipconfig command is a network configuration utility available in Windows. It is used to display and manage the IP address, subnet mask, default gateway, and other network settings of a device.

Ex:

C:\Users\HEMANTH>ipconfig



```
Command Prompt
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH>ipconfig

Windows IP Configuration

Wireless LAN adapter Local Area Connection* 1:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 2:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi:

    Connection-specific DNS Suffix  . :
    IPv6 Address. . . . . : fd01::813a:2240:4851:47d8
    Temporary IPv6 Address. . . . . : fd01::95ea:de00:636b:a410
    Link-local IPv6 Address . . . . . : fe80::51c1:90df:a0a4:c8f7%12
    IPv4 Address. . . . . : 192.168.0.113
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : fe80::f2b4:d2ff:fe60:d963%12
                                192.168.0.1

Ethernet adapter Bluetooth Network Connection:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

C:\Users\HEMANTH>
```

ipconfig /all : The ipconfig /all command in Windows provides a comprehensive view of the network configuration for all network interfaces on the system. It displays detailed

information, such as the IP address, subnet mask, default gateway, and other key network settings, including DNS servers and DHCP status.

Ex:

C:\Users\HEMANTH>ipconfig/all

```

Microsoft Windows [Version 10.0.26100.3620]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH>ipconfig/all

Windows IP Configuration

Host Name . . . . . : LAPTOP-HJ3VQ0LI
Primary Dns Suffix . . . . . : 
Node Type . . . . . : Mixed
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No

Wireless LAN adapter Local Area Connection* 1:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . : 
Description . . . . . : Microsoft Wi-Fi Direct Virtual Adapter
Physical Address. . . . . : A0-80-69-E9-08-FE
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes

Wireless LAN adapter Local Area Connection* 2:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . : 
Description . . . . . : Microsoft Wi-Fi Direct Virtual Adapter #2
Physical Address. . . . . : A2-80-69-E9-08-FE
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . : Yes

Wireless LAN adapter Wi-Fi:

Connection-specific DNS Suffix . : 
Description . . . . . : Intel(R) Wi-Fi 6E AX211 160MHz
Physical Address. . . . . : A0-80-69-E9-08-FE
Autoconfiguration Enabled . . . . : Yes
IPv6 Address. . . . . : fd81::813a:2240:4851:07d8(Preferred)
Temporary IPv6 Address. . . . . : fd81::96ea:da00:63db:a410(Preferred)
Link-local IPv6 Address . . . . : fe80::31c1:90d9:a9a1:c67a12(Preferred)
IPv4 Address. . . . . : 192.168.0.113(Preferred)
Subnet Mask . . . . . : 255.255.255.0
Lease Obtained. . . . . : 02 April 2025 06:03:07 PM
Lease Expires . . . . . : 03 April 2025 06:03:06 PM
Default Gateway . . . . . : fe80::f2b4:d2ff:fe68:d963%12
192.168.0.1
DHCP Server . . . . . : 192.168.0.1
DHCPv6 Iaid . . . . . : 248339657
DHCPv6 Client DUID. . . . . : 00-01-00-01-2E-EF-63-5F-A0-80-69-E9-08-FE
DNS Servers . . . . . : fe80::f2b4:d2ff:fe68:d963%12
192.168.0.1
NetBIOS over Tcpip. . . . . : Enabled

Ethernet adapter Bluetooth Network Connection:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . : 
Description . . . . . : Bluetooth Device (Personal Area Network)
Physical Address. . . . . : A0-80-69-E9-09-02
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes

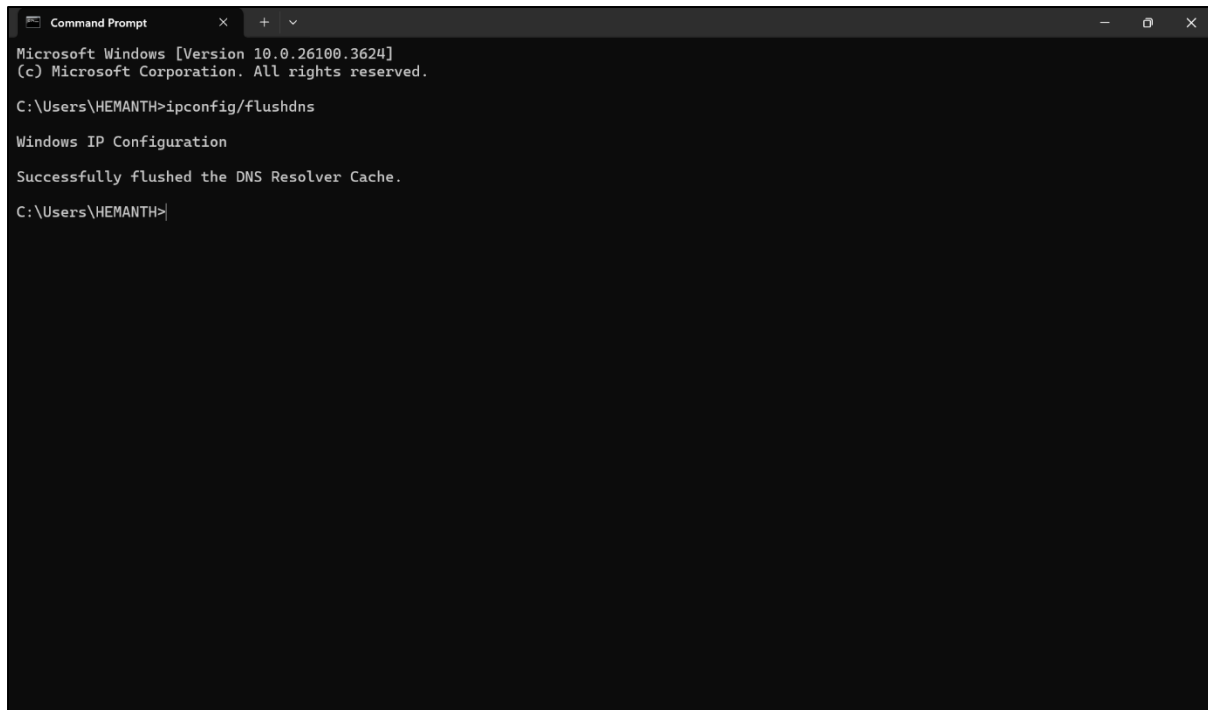
C:\Users\HEMANTH>

```

ipconfig /flushdns: The ipconfig /flushdns command is used in Windows to clear the DNS resolver cache, which stores previously resolved domain names to IP addresses. This helps in resolving DNS-related issues by forcing the system to retrieve fresh DNS information.

Ex:

C:\Users\HEMANTH>ipconfig/flushdns



```
Command Prompt
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH>ipconfig /flushdns

Windows IP Configuration

Successfully flushed the DNS Resolver Cache.

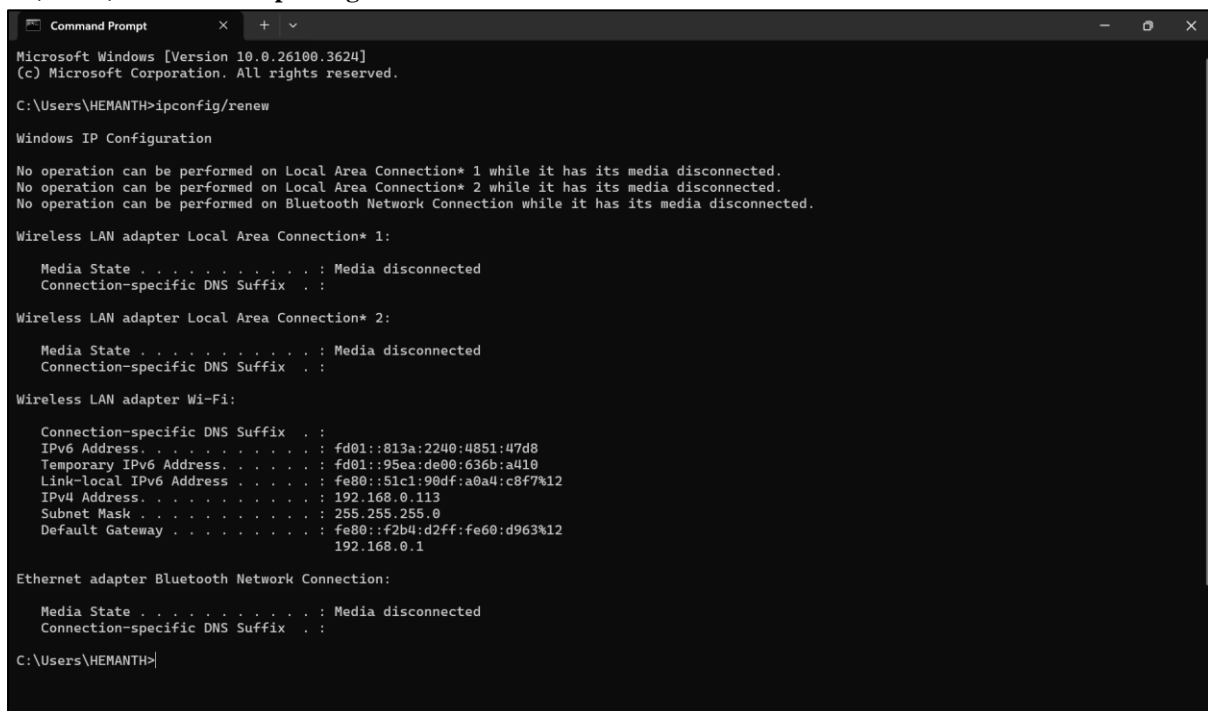
C:\Users\HEMANTH>
```

ipconfig /renew:

The ipconfig /renew command in Windows is used to request a new IP address from the DHCP server for a network adapter. This is useful when troubleshooting network issues, especially if you are experiencing connectivity problems.

Ex:

C:\Users\HEMANTH>ipconfig/renew



```
Command Prompt
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH>ipconfig /renew

Windows IP Configuration

No operation can be performed on Local Area Connection* 1 while it has its media disconnected.
No operation can be performed on Local Area Connection* 2 while it has its media disconnected.
No operation can be performed on Bluetooth Network Connection while it has its media disconnected.

Wireless LAN adapter Local Area Connection* 1:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix . :

Wireless LAN adapter Local Area Connection* 2:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix . :

Wireless LAN adapter Wi-Fi:

    Connection-specific DNS Suffix . :
    IPv6 Address. . . . . : fd01::813a:2240:4851:47d8
    Temporary IPv6 Address. . . . . : fd01::95ea:de00:636b:a410
    Link-local IPv6 Address . . . . . : fe80::51c1:90df:a0a4:c8f7%12
    IPv4 Address. . . . . : 192.168.0.113
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : fe80::f2b4:d2ff:fe60:d963%12
                                192.168.0.1

Ethernet adapter Bluetooth Network Connection:

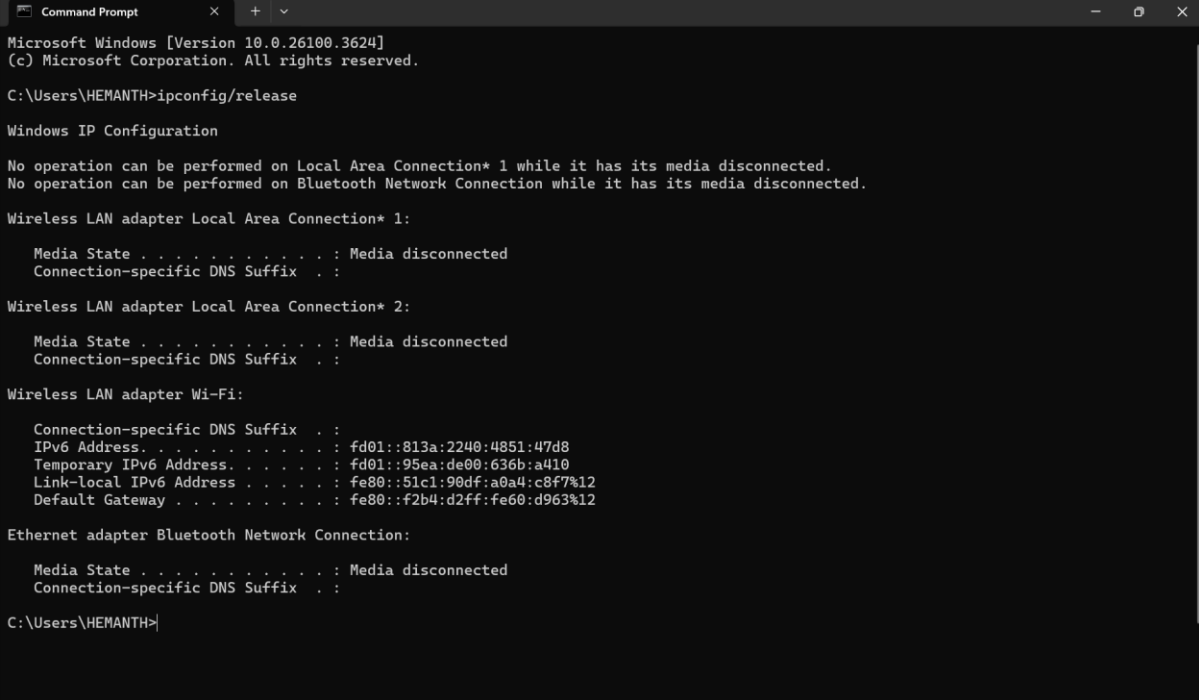
    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix . :

C:\Users\HEMANTH>
```

ipconfig /release: The ipconfig /release command is used to release the current IP address assigned by the DHCP server. This is useful when troubleshooting network issues or switching to a new network.

Ex:

C:\Users\HEMANTH>ipconfig/release



```

Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH>ipconfig/release

Windows IP Configuration

No operation can be performed on Local Area Connection* 1 while it has its media disconnected.
No operation can be performed on Bluetooth Network Connection while it has its media disconnected.

Wireless LAN adapter Local Area Connection* 1:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 2:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi:

    Connection-specific DNS Suffix  . :
    IPv6 Address. . . . . : fd01::813a:2240:4851:47d8
    Temporary IPv6 Address. . . . . : fd01::95ea:de00:636b:a410
    Link-local IPv6 Address . . . . . : fe80::51c1:90df:a0a4:c8f7%12
    Default Gateway . . . . . : fe80::f2b4:d2ff:fe60:d963%12

Ethernet adapter Bluetooth Network Connection:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

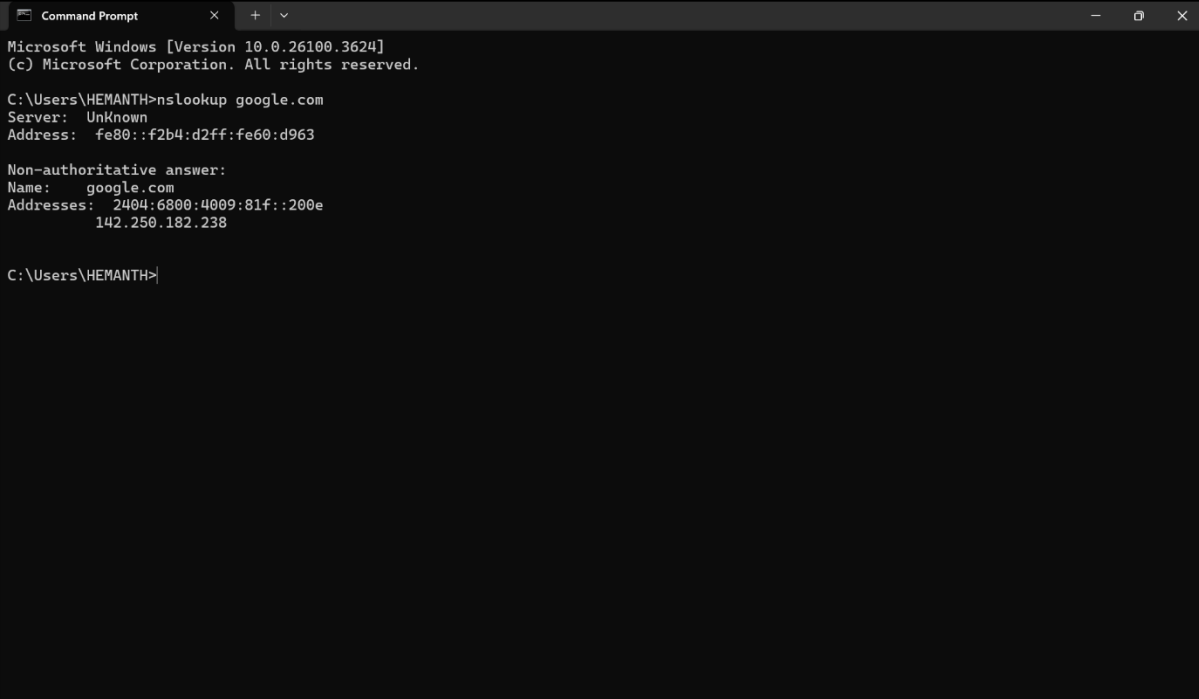
C:\Users\HEMANTH>

```

Nslookup: The nslookup (Name Server Lookup) command is used to query the Domain Name System (DNS) to obtain domain name or IP address mapping details. It is useful for troubleshooting DNS-related issues and checking domain records.

Ex:

C:\Users\HEMANTH>nslookup google.com



```

Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH>nslookup google.com
Server:  UnKnown
Address:  fe80::f2b4:d2ff:fe60:d963

Non-authoritative answer:
Name:    google.com
Addresses:  2404:6800:4009:81f::200e
           142.250.182.238

C:\Users\HEMANTH>

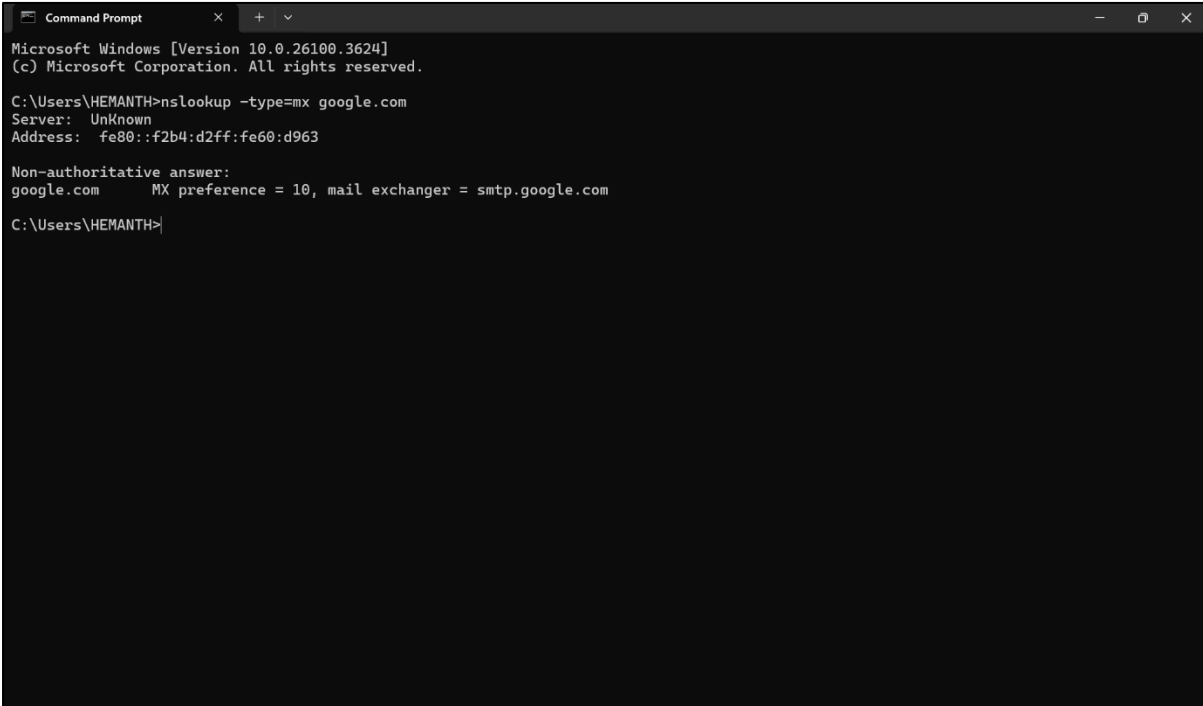
```

-type=<record>: In nslookup, the -type=<record> option is used to specify the type of DNS record you want to query. This allows you to retrieve different types of DNS

information, such as A (IPv4), AAAA (IPv6), MX (Mail Exchange), TXT (Text), and more.

Ex:

C:\Users\HEMANTH>nslookup -type=mx google.com



```
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH>nslookup -type=mx google.com
Server:      UnKnown
Address:     fe80::f2b4:d2ff:fe60:d963

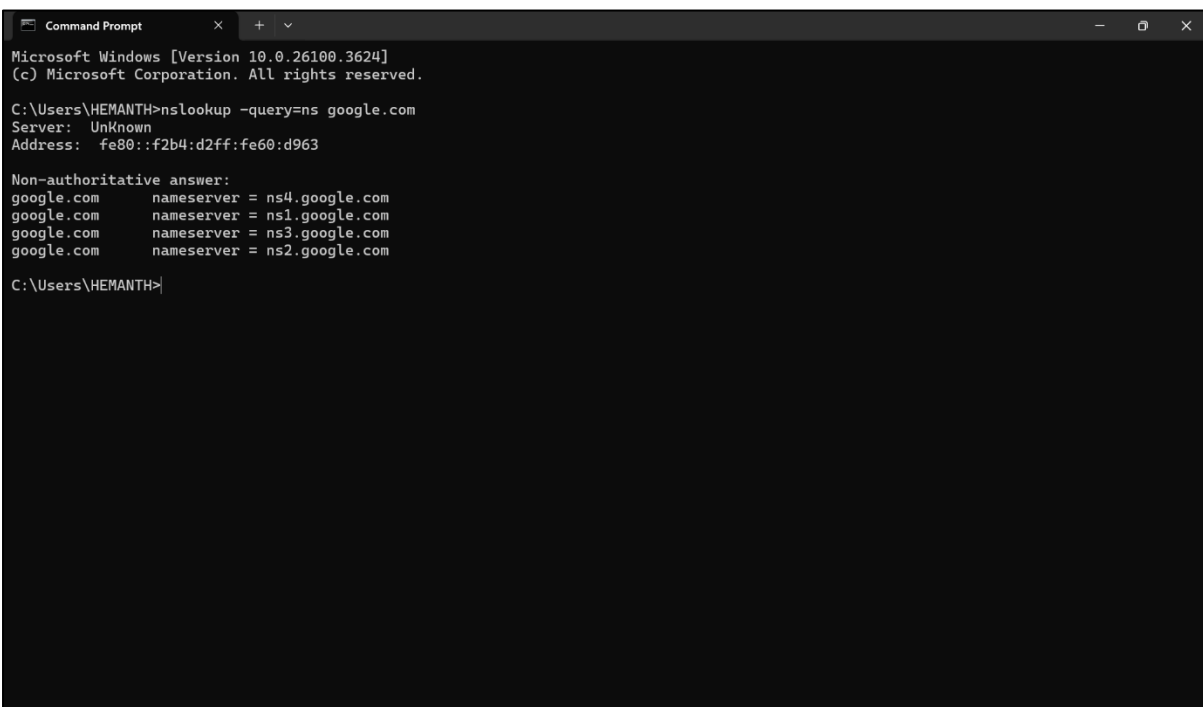
Non-authoritative answer:
google.com   MX preference = 10, mail exchanger = smtp.google.com

C:\Users\HEMANTH>
```

-query=<record>: The -query=<record> option in nslookup is an alternative way to specify the type of DNS record you want to retrieve. It works the same way as -type=<record> and allows you to query different DNS record types.

Ex:

C:\Users\HEMANTH>nslookup -query=ns google.com



```
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH>nslookup -query=ns google.com
Server:      UnKnown
Address:     fe80::f2b4:d2ff:fe60:d963

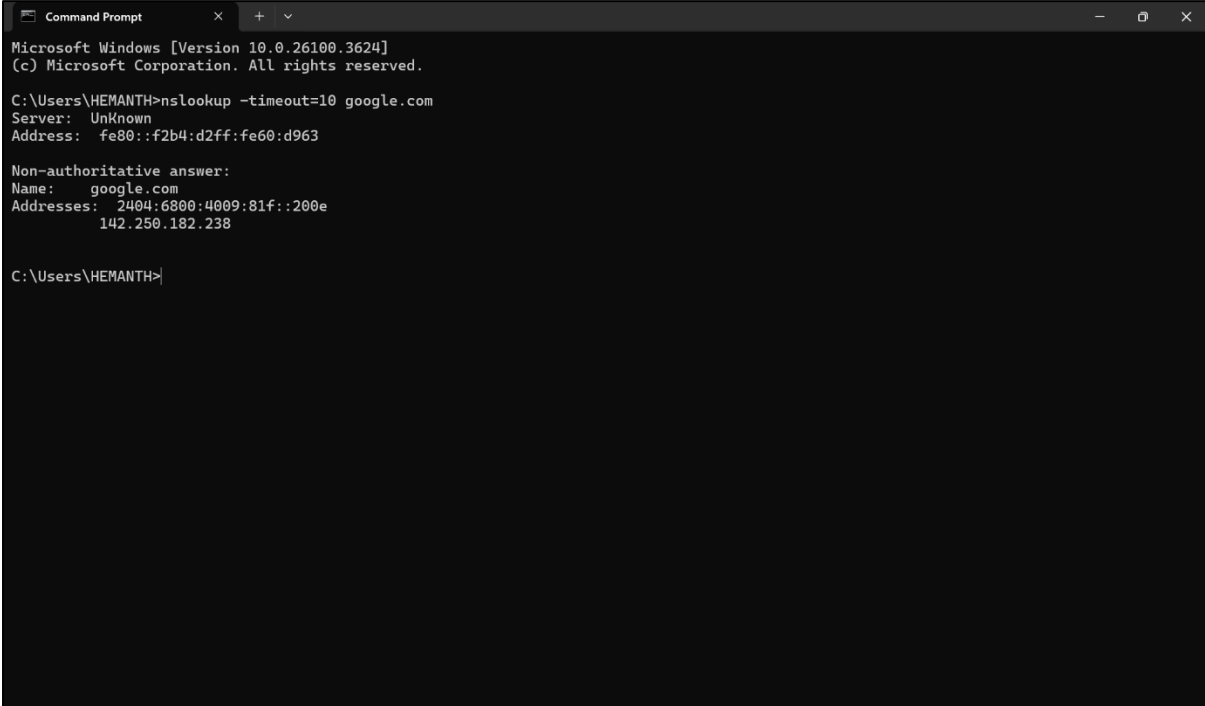
Non-authoritative answer:
google.com   nameserver = ns4.google.com
google.com   nameserver = ns1.google.com
google.com   nameserver = ns3.google.com
google.com   nameserver = ns2.google.com

C:\Users\HEMANTH>
```

-timeout<seconds>:The -timeout=<seconds> option in nslookup is used to specify the number of seconds the command should wait for a response from a DNS server before timing out. This is useful when troubleshooting slow or unresponsive DNS queries.

Ex:

C:\Users\HEMANTH>nslookup -timeout=10 google.com



```
Command Prompt
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH>nslookup -timeout=10 google.com
Server:      UnKnown
Address:     fe80::f2b4:d2ff:fe60:d963

Non-authoritative answer:
Name:       google.com
Addresses:  2404:6800:4009:81f::200e
           142.250.182.238

C:\Users\HEMANTH>
```

-retry=<count>:The -retry=<count> option in nslookup specifies the number of times the command will attempt to contact the DNS server before giving up if it does not receive a response. This is useful for handling intermittent network issues or slow DNS responses.

Ex:

C:\Users\HEMANTH>nslookup -retry=3 google.com

```

Command Prompt
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH>nslookup -retry=3 google.com
Server:      UnKnown
Address:     fe80::f2b4:d2ff:fe60:d963

Non-authoritative answer:
Name:       google.com
Addresses:  2404:6800:4009:81f::200e
            142.250.182.238

C:\Users\HEMANTH>

```

netstat :

netstat is a command-line tool used to display network connections, routing tables, interface statistics, and network protocol information. It helps monitor and troubleshoot network-related issues on a system.

Ex:

C:\Users\HEMANTH>netstat

```

Command Prompt
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH>netstat
Active Connections

Proto Local Address           Foreign Address         State
TCP    127.0.0.1:1638           LAPTOP-KJ3VQ011:1638   ESTABLISHED
TCP    127.0.0.1:1639           LAPTOP-KJ3VQ011:1638   ESTABLISHED
TCP    127.0.0.1:1640           LAPTOP-KJ3VQ011:1641   ESTABLISHED
TCP    127.0.0.1:1641           LAPTOP-KJ3VQ011:1640   ESTABLISHED
TCP    127.0.0.1:1681           LAPTOP-KJ3VQ011:1682   ESTABLISHED
TCP    127.0.0.1:1682           LAPTOP-KJ3VQ011:1681   ESTABLISHED
TCP    127.0.0.1:1683           LAPTOP-KJ3VQ011:1684   ESTABLISHED
TCP    127.0.0.1:1684           LAPTOP-KJ3VQ011:1683   ESTABLISHED
TCP    127.0.0.1:1685           LAPTOP-KJ3VQ011:1685   ESTABLISHED
TCP    127.0.0.1:1686           LAPTOP-KJ3VQ011:1685   ESTABLISHED
TCP    127.0.0.1:1686           LAPTOP-KJ3VQ011:1689   ESTABLISHED
TCP    127.0.0.1:1689           LAPTOP-KJ3VQ011:1688   ESTABLISHED
TCP    192.168.0.113:1025       4.213.25.241:https     ESTABLISHED
TCP    192.168.0.113:8872      192.168.0.109:8089     ESTABLISHED
TCP    192.168.0.113:8874      4.213.25.241:https     ESTABLISHED
TCP    192.168.0.113:8960      sh-in-f108:8228        ESTABLISHED
TCP    192.168.0.113:9069      104.18.32.47:https     ESTABLISHED
TCP    192.168.0.113:9016      104.18.32.47:https     ESTABLISHED
TCP    192.168.0.113:9080      158.177.84.254:https   CLOSE_WAIT
TCP    192.168.0.113:9081      13.107.246.254:https   CLOSE_WAIT
TCP    192.168.0.113:9094      bom07s97-in-f14:https  CLOSE_WAIT
TCP    192.168.0.113:9111      13.107.226.58:https    CLOSE_WAIT
TCP    192.168.0.113:9112      13.107.246.254:https   CLOSE_WAIT
TCP    192.168.0.113:9117      ec2-3-111-224-186:https ESTABLISHED
TCP    192.168.0.113:9128      49.44.198.236:https    LAST_ACK
TCP    192.168.0.113:9132      204.79.197.222:https   ESTABLISHED
TCP    192.168.0.113:9134      a4de5b0d0665299a:http  TIME_WAIT
TCP    192.168.0.113:9135      whatsapp-cdn-shv-03-bom2:https CLOSE_WAIT
TCP    192.168.0.113:9136      whatsapp-cdn-shv-02-maa2:https CLOSE_WAIT
TCP    192.168.0.113:9137      static-292-60-139-224:https ESTABLISHED
TCP    192.168.0.113:9138      whatsapp-cdn-shv-03-bom2:https CLOSE_WAIT
TCP    192.168.0.113:9139      static-182:https       ESTABLISHED
TCP    192.168.0.113:9140      13.69.239.73:https     ESTABLISHED
TCP    192.168.0.113:9142      13.105.28.16:https     ESTABLISHED
TCP    192.168.0.113:9143      52.239.235.196:https   ESTABLISHED
TCP    192.168.0.113:9146      52.123.129.16:https    ESTABLISHED
TCP    192.168.0.113:9147      52.123.129.16:https    ESTABLISHED
TCP    192.168.0.113:9148      49.44.196.225:https    ESTABLISHED
TCP    192.168.0.113:9149      a-0803:https           TIME_WAIT
TCP    192.168.0.113:9154      49.44.196.162:https    ESTABLISHED
TCP    192.168.0.113:9156      49.44.198.236:https    ESTABLISHED
TCP    192.168.0.113:9157      bingforbusiness:https  ESTABLISHED
TCP    192.168.0.113:9158      49.44.183.34:https     ESTABLISHED
TCP    192.168.0.113:9159      49.44.183.34:https     ESTABLISHED
TCP    192.168.0.113:9160      49.44.183.34:https     ESTABLISHED
TCP    192.168.0.113:9161      49.44.183.34:https     ESTABLISHED
TCP    192.168.0.113:9162      49.44.183.34:https     ESTABLISHED
TCP    192.168.0.113:9163      49.44.183.34:https     ESTABLISHED
TCP    192.168.0.113:9164      20.195.84.16:https     ESTABLISHED
TCP    192.168.0.113:9165      20.195.84.16:https     ESTABLISHED
TCP    192.168.0.113:9166      bingforbusiness:https  ESTABLISHED
TCP    [F691::F69a:de00:636:a19]:9167 whatsapp-cdn-shv-02-bom1:http SYN_SENT

C:\Users\HEMANTH>

```

-t: The **-t** option in the **netstat** command is used to display TCP (Transmission Control Protocol) connections. When you run **netstat -t**, it will show you a list of all active TCP

connections on your system, along with their current states.

Ex:

C:\Users\HEMANTH>netstat -t

```

Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH>netstat -t

Active Connections

Proto Local Address           Foreign Address         State       Offload State
TCP    127.0.0.1:1038           LAPTOP-KJ3VQ011:1038  ESTABLISHED InHost
TCP    127.0.0.1:1039           LAPTOP-KJ3VQ011:1038  ESTABLISHED InHost
TCP    127.0.0.1:1040           LAPTOP-KJ3VQ011:1041  ESTABLISHED InHost
TCP    127.0.0.1:1041           LAPTOP-KJ3VQ011:1040  ESTABLISHED InHost
TCP    127.0.0.1:1081           LAPTOP-KJ3VQ011:1082  ESTABLISHED InHost
TCP    127.0.0.1:1082           LAPTOP-KJ3VQ011:1081  ESTABLISHED InHost
TCP    127.0.0.1:1083           LAPTOP-KJ3VQ011:1084  ESTABLISHED InHost
TCP    127.0.0.1:1084           LAPTOP-KJ3VQ011:1083  ESTABLISHED InHost
TCP    127.0.0.1:1085           LAPTOP-KJ3VQ011:1086  ESTABLISHED InHost
TCP    127.0.0.1:1086           LAPTOP-KJ3VQ011:1085  ESTABLISHED InHost
TCP    127.0.0.1:1088           LAPTOP-KJ3VQ011:1089  ESTABLISHED InHost
TCP    127.0.0.1:1089           LAPTOP-KJ3VQ011:1088  ESTABLISHED InHost
TCP    192.168.0.113:1025       4.213.25.241:https    ESTABLISHED InHost
TCP    192.168.0.113:8872       192.168.0.109:8009    ESTABLISHED InHost
TCP    192.168.0.113:8874       4.213.25.241:https    ESTABLISHED InHost
TCP    192.168.0.113:8960       sh-in-f188:5228       ESTABLISHED InHost
TCP    192.168.0.113:9086       158.171.28.254:https  CLOSE_WAIT InHost
TCP    192.168.0.113:9081       13.107.246.254:https  CLOSE_WAIT InHost
TCP    192.168.0.113:9111       13.107.226.58:https   CLOSE_WAIT InHost
TCP    192.168.0.113:9112       13.107.246.254:https  CLOSE_WAIT InHost
TCP    192.168.0.113:9117       ec2-3-111-224-186:https ESTABLISHED InHost
TCP    192.168.0.113:9152       204.79.197.222:https  FIN_WAIT_1 InHost
TCP    192.168.0.113:9191       20.195.84.16:https    ESTABLISHED InHost
TCP    192.168.0.113:9204       104.18.32.47:https    ESTABLISHED InHost
TCP    192.168.0.113:9205       192.168.0.113:9205    TIME_WAIT InHost
TCP    192.168.0.113:9207       1:https               TIME_WAIT InHost
TCP    192.168.0.113:9210       172.64.155.209:https  ESTABLISHED InHost
TCP    192.168.0.113:9211       172.64.155.209:https  TIME_WAIT InHost
TCP    192.168.0.113:9213       a23-53-248-177:https  ESTABLISHED InHost
TCP    192.168.0.113:9216       ec2-13-203-1-146:https ESTABLISHED InHost
TCP    192.168.0.113:9218       172.64.155.209:https  TIME_WAIT InHost
TCP    192.168.0.113:9220       1:https               ESTABLISHED InHost
TCP    192.168.0.113:9227       216.24.57.252:https  ESTABLISHED InHost
TCP    192.168.0.113:9229       216.24.57.252:https  ESTABLISHED InHost
TCP    192.168.0.113:9230       server-108-157-238-42:https ESTABLISHED InHost
TCP    192.168.0.113:9231       server-18-155-33-63:https ESTABLISHED InHost
TCP    192.168.0.113:9232       ec2-18-235-47-239:https CLOSE_WAIT InHost
TCP    192.168.0.113:9234       ec2-35-176-127-31:https ESTABLISHED InHost
TCP    192.168.0.113:9237       bom12614-in-f18:https ESTABLISHED InHost
TCP    192.168.0.113:9238       bom12614-in-f18:https ESTABLISHED InHost
TCP    192.168.0.113:9240       ec2-3-233-158-24:https ESTABLISHED InHost
TCP    192.168.0.113:9241       49.44.198.238:https   ESTABLISHED InHost
TCP    192.168.0.113:9242       49.44.183.34:https    ESTABLISHED InHost
TCP    192.168.0.113:9243       49.44.198.234:https   ESTABLISHED InHost
TCP    192.168.0.113:9244       49.44.183.34:https    ESTABLISHED InHost
TCP    192.168.0.113:9245       52.182.143.213:https  ESTABLISHED InHost
TCP    192.168.0.113:9246       13.107.138.254:https  ESTABLISHED InHost
TCP    192.168.0.113:9247       158.171.31.254:https  ESTABLISHED InHost
TCP    192.168.0.113:9248       4.159.249.254:https   ESTABLISHED InHost
TCP    192.168.0.113:9249       204.79.197.222:https  ESTABLISHED InHost
C:\Users\HEMANTH>

```

-n: The -n option in netstat displays network connections using numerical IP addresses and port numbers instead of resolving them to hostnames and service names.

Ex:

C:\Users\HEMANTH>netstat -n

```

Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH>netstat -n

Active Connections

Proto Local Address           Foreign Address         State
TCP    127.0.0.1:1038           127.0.0.1:1039         ESTABLISHED
TCP    127.0.0.1:1039           127.0.0.1:1038         ESTABLISHED
TCP    127.0.0.1:1040           127.0.0.1:1041         ESTABLISHED
TCP    127.0.0.1:1041           127.0.0.1:1040         ESTABLISHED
TCP    127.0.0.1:1081           127.0.0.1:1082         ESTABLISHED
TCP    127.0.0.1:1082           127.0.0.1:1081         ESTABLISHED
TCP    127.0.0.1:1083           127.0.0.1:1084         ESTABLISHED
TCP    127.0.0.1:1084           127.0.0.1:1083         ESTABLISHED
TCP    127.0.0.1:1085           127.0.0.1:1086         ESTABLISHED
TCP    127.0.0.1:1086           127.0.0.1:1085         ESTABLISHED
TCP    127.0.0.1:1088           127.0.0.1:1089         ESTABLISHED
TCP    127.0.0.1:1089           127.0.0.1:1088         ESTABLISHED
TCP    192.168.0.113:1025       4.213.25.241:443       ESTABLISHED
TCP    192.168.0.113:8872       192.168.0.109:8009     ESTABLISHED
TCP    192.168.0.113:8874       4.213.25.241:443       ESTABLISHED
TCP    192.168.0.113:8960       102.251.178.188:5228   ESTABLISHED
TCP    192.168.0.113:9191       20.195.84.16:443       ESTABLISHED
TCP    192.168.0.113:9216       13.203.1.146:443       CLOSE_WAIT
TCP    192.168.0.113:9220       35.190.80.1:443        ESTABLISHED
TCP    192.168.0.113:9224       35.170.127.31:443      ESTABLISHED
TCP    192.168.0.113:9242       49.44.183.34:443       LAST_ACK
TCP    192.168.0.113:9243       49.44.198.234:443      LAST_ACK
TCP    192.168.0.113:9256       13.203.1.146:443       ESTABLISHED
TCP    192.168.0.113:9258       52.109.56.129:443      TIME_WAIT
TCP    192.168.0.113:9259       13.107.137.11:443      ESTABLISHED
TCP    192.168.0.113:9260       52.104.116.41:443      ESTABLISHED
TCP    192.168.0.113:9261       20.42.65.88:443        ESTABLISHED
TCP    192.168.0.113:9262       13.107.137.11:443      ESTABLISHED
TCP    192.168.0.113:9263       13.107.137.11:443      ESTABLISHED
TCP    192.168.0.113:9264       13.107.137.11:443      ESTABLISHED
TCP    192.168.0.113:9266       13.107.137.11:443      ESTABLISHED
TCP    192.168.0.113:9267       13.107.137.11:443      ESTABLISHED
TCP    192.168.0.113:9268       13.107.137.11:443      ESTABLISHED
TCP    192.168.0.113:9269       13.107.137.11:443      ESTABLISHED
TCP    192.168.0.113:9270       13.107.137.11:443      ESTABLISHED
TCP    192.168.0.113:9271       49.44.158.234:443      ESTABLISHED
TCP    192.168.0.113:9272       52.182.143.213:443     ESTABLISHED
TCP    192.168.0.113:9273       158.171.28.254:443     ESTABLISHED
TCP    192.168.0.113:9274       131.253.33.254:443     ESTABLISHED
TCP    192.168.0.113:9275       13.107.4.254:443       ESTABLISHED
TCP    192.168.0.113:9276       204.79.197.222:443     ESTABLISHED
C:\Users\HEMANTH>

```

-o: netstat -o shows active network connections along with the Process ID (PID), helping identify which process is using a specific port.

Ex:

C:\Users\HEMANTH>netstat -o

```

Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH>netstat -o

Active Connections

Proto Local Address           Foreign Address         State       PID
TCP    127.0.0.1:1038           LAPTOP-WJ3VQD1I:1039  ESTABLISHED 7280
TCP    127.0.0.1:1039           LAPTOP-WJ3VQD1I:1038  ESTABLISHED 7280
TCP    127.0.0.1:1040           LAPTOP-WJ3VQD1I:1041  ESTABLISHED 7280
TCP    127.0.0.1:1041           LAPTOP-WJ3VQD1I:1040  ESTABLISHED 7280
TCP    127.0.0.1:1081           LAPTOP-WJ3VQD1I:1082  ESTABLISHED 1908
TCP    127.0.0.1:1082           LAPTOP-WJ3VQD1I:1081  ESTABLISHED 1908
TCP    127.0.0.1:1083           LAPTOP-WJ3VQD1I:1084  ESTABLISHED 2064
TCP    127.0.0.1:1084           LAPTOP-WJ3VQD1I:1083  ESTABLISHED 2064
TCP    127.0.0.1:1085           LAPTOP-WJ3VQD1I:1086  ESTABLISHED 5220
TCP    127.0.0.1:1086           LAPTOP-WJ3VQD1I:1085  ESTABLISHED 5220
TCP    127.0.0.1:1088           LAPTOP-WJ3VQD1I:1089  ESTABLISHED 3760
TCP    127.0.0.1:1089           LAPTOP-WJ3VQD1I:1088  ESTABLISHED 3760
TCP    192.168.0.113:1025       4.213.25.241:https    ESTABLISHED 5664
TCP    192.168.0.113:8072      192.168.0.109:8009    ESTABLISHED 17720
TCP    192.168.0.113:8074      4.213.25.241:https    ESTABLISHED 19928
TCP    192.168.0.113:8900      sh-in-f188:5228       ESTABLISHED 17720
TCP    192.168.0.113:9191      20.195.84.16:https    ESTABLISHED 3732
TCP    192.168.0.113:9220      1:https               ESTABLISHED 17720
TCP    192.168.0.113:9234      ec2-35-174-127-31:https ESTABLISHED 17720
TCP    192.168.0.113:9256      ec2-13-203-1-146:https TIME_WAIT 0
TCP    192.168.0.113:9260      52.104.116.41:https   TIME_WAIT 0
TCP    192.168.0.113:9261      20.42.65.88:https     TIME_WAIT 0
TCP    192.168.0.113:9272      52.102.143.213:https  TIME_WAIT 0
TCP    192.168.0.113:9273      150.171.28.254:https  ESTABLISHED 14608
TCP    192.168.0.113:9274      131.253.33.254:https  ESTABLISHED 14608
TCP    192.168.0.113:9275      13.107.4.254:https    ESTABLISHED 14608
TCP    192.168.0.113:9276      204.79.197.222:https  TIME_WAIT 0
TCP    192.168.0.113:9277      13.107.197.11:https   ESTABLISHED 19928
TCP    192.168.0.113:9278      52.104.116.41:https   ESTABLISHED 19928
TCP    192.168.0.113:9279      20.42.65.88:https     ESTABLISHED 19928
TCP    192.168.0.113:9280      49.44.198.234:https   ESTABLISHED 14608
TCP    192.168.0.113:9281      52.102.143.213:https  ESTABLISHED 14608
TCP    192.168.0.113:9282      150.171.70.254:https  ESTABLISHED 14608
TCP    192.168.0.113:9283      13.107.246.254:https  TIME_WAIT 0
TCP    192.168.0.113:9284      13.107.246.254:https  ESTABLISHED 14608
TCP    192.168.0.113:9285      204.79.197.222:https  ESTABLISHED 14608
TCP    192.168.0.113:9288      ec2-13-203-1-146:https ESTABLISHED 5560
TCP    192.168.0.113:9289      13.107.213.254:https  ESTABLISHED 14608
TCP    192.168.0.113:9290      13.107.6.254:https    ESTABLISHED 14608
TCP    192.168.0.113:9291      128.251.95.9:https    ESTABLISHED 14608

C:\Users\HEMANTH>

```

-e: The -e option in the netstat command is used to display network interface statistics, including the number of packets sent and received, errors, and dropped packets.

Ex:

C:\Users\HEMANTH>netstat -e

```

Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH>netstat -e

Interface Statistics

           Received           Sent
Bytes      260547984      144542406
Unicast packets      336306      243390
Non-unicast packets      966      44310
Discards      0      0
Errors      0      0
Unknown protocols      0      0

C:\Users\HEMANTH>

```


Experiment-2

AIM: To implement CRC for error detection using the CRC-CCIT generator.

Description: Cyclic Redundancy Check (CRC) is a method used to detect errors in digital data. The **CRC-CCITT** generator is a standard polynomial used in many communication systems.

Sender:

```
import java.util.Scanner;

public class Sender1 {

    static final String POLYNOMIAL = "100000111";
    static final int POLY_LENGTH = POLYNOMIAL.length();

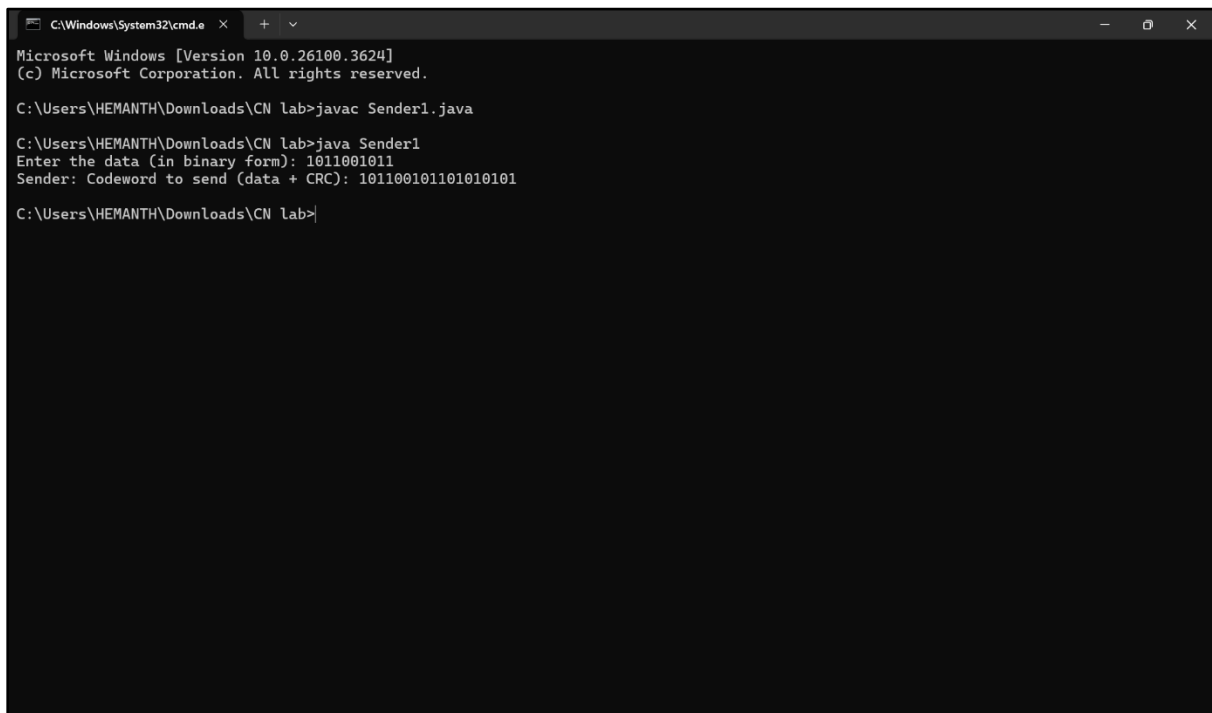
    static String xor(String a, String b) {
        StringBuilder result = new StringBuilder();
        for (int i = 0; i < a.length(); i++) {
            result.append(a.charAt(i) == b.charAt(i) ? '0' : '1');
        }
        return result.toString();
    }

    static String calculateCRC(String data) {
        StringBuilder dataWithZeros = new StringBuilder(data);
        for (int i = 0; i < POLY_LENGTH - 1; i++) {
            dataWithZeros.append('0');
        }
        String remainder = dataWithZeros.toString();
        for (int i = 0; i < data.length(); i++) {
            if (remainder.charAt(i) == '1') {
                remainder = remainder.substring(0, i) + xor(remainder.substring(i, i + POLY_LENGTH),
POLYNOMIAL) + remainder.substring(i + POLY_LENGTH);
            }
        }
        return remainder.substring(data.length(), remainder.length());
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the data (in binary form): ");
        String data = scanner.nextLine();
        if (!data.matches("[01]+")) {
            System.out.println("Invalid input. Please enter a binary string.");
        }
    }
}
```

```
        return;  
    }  
    String crc = calculateCRC(data);  
    String codeword = data + crc;  
    System.out.println("Sender: Codeword to send (data + CRC): " + codeword);  
    scanner.close();  
}  
}
```

Output:



```
C:\Windows\System32\cmd.e x + v  
Microsoft Windows [Version 10.0.26100.3624]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\HEMANTH\Downloads\CN lab>javac Sender1.java  
  
C:\Users\HEMANTH\Downloads\CN lab>java Sender1  
Enter the data (in binary form): 1011001011  
Sender: Codeword to send (data + CRC): 101100101101010101  
  
C:\Users\HEMANTH\Downloads\CN lab>
```

Receiver:

```
import java.util.Scanner;

public class Receiver1 {

    static final String POLYNOMIAL = "100000111";
    static final int POLY_LENGTH = POLYNOMIAL.length();

    static String xor(String a, String b) {
        StringBuilder result = new StringBuilder();
        for (int i = 0; i < a.length(); i++) {
            result.append(a.charAt(i) == b.charAt(i) ? '0' : '1');
        }
        return result.toString();
    }

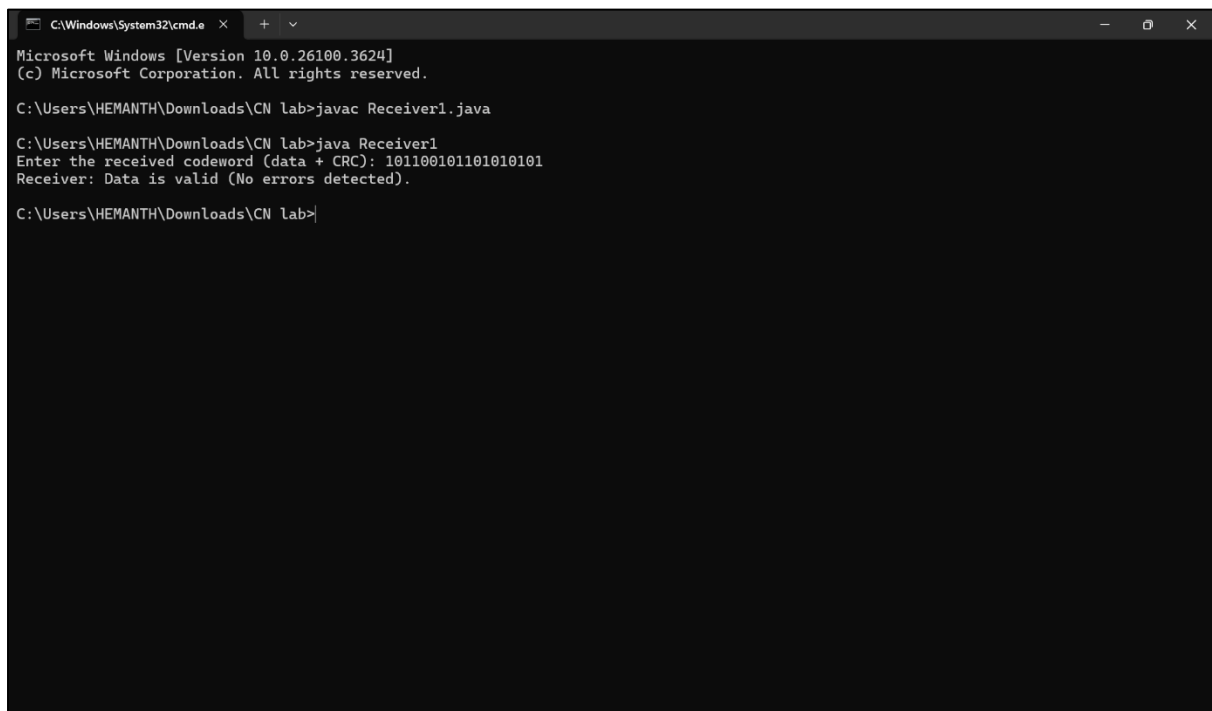
    static String calculateCRC(String data) {
        StringBuilder dataWithZeros = new StringBuilder(data);
        for (int i = 0; i < POLY_LENGTH - 1; i++) {
            dataWithZeros.append('0');
        }
        String remainder = dataWithZeros.toString();
        for (int i = 0; i < data.length(); i++) {
            if (remainder.charAt(i) == '1') {
                remainder = remainder.substring(0, i) + xor(remainder.substring(i, i + POLY_LENGTH),
POLYNOMIAL) + remainder.substring(i + POLY_LENGTH);
            }
        }
        return remainder.substring(data.length(), remainder.length());
    }

    static boolean checkCRC(String dataWithCRC) {
        String crc = calculateCRC(dataWithCRC);
        return crc.equals("00000000");
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the received codeword (data + CRC): ");
        String receivedData = scanner.nextLine();
        if (!receivedData.matches("[01]+")) {
            System.out.println("Invalid input. Please enter a binary string.");
            return;
        }
    }
}
```

```
    }  
    if (checkCRC(receivedData)) {  
        System.out.println("Receiver: Data is valid (No errors detected).");  
    } else {  
        System.out.println("Receiver: Data is invalid (Errors detected).");  
    }  
    scanner.close();  
}  
}
```

Output:



```
C:\Windows\System32\cmd.exe  
Microsoft Windows [Version 10.0.26100.3624]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\HEMANTH\Downloads\CN lab>javac Receiver1.java  
  
C:\Users\HEMANTH\Downloads\CN lab>java Receiver1  
Enter the received codeword (data + CRC): 101100101101010101  
Receiver: Data is valid (No errors detected).  
  
C:\Users\HEMANTH\Downloads\CN lab>
```

Experiment-3

AIM: To implement data-link layer sharing for bit-stuffing.

Description: Bit stuffing is a technique used in data communication to prevent confusion with special control sequences. When sending data, a '0' is inserted after five consecutive '1's in the data stream. This ensures that special patterns (like frame delimiters) are not accidentally interpreted as part of the data. At the receiver's end, the stuffed '0's are removed to recover the original data.

BitStuffing:

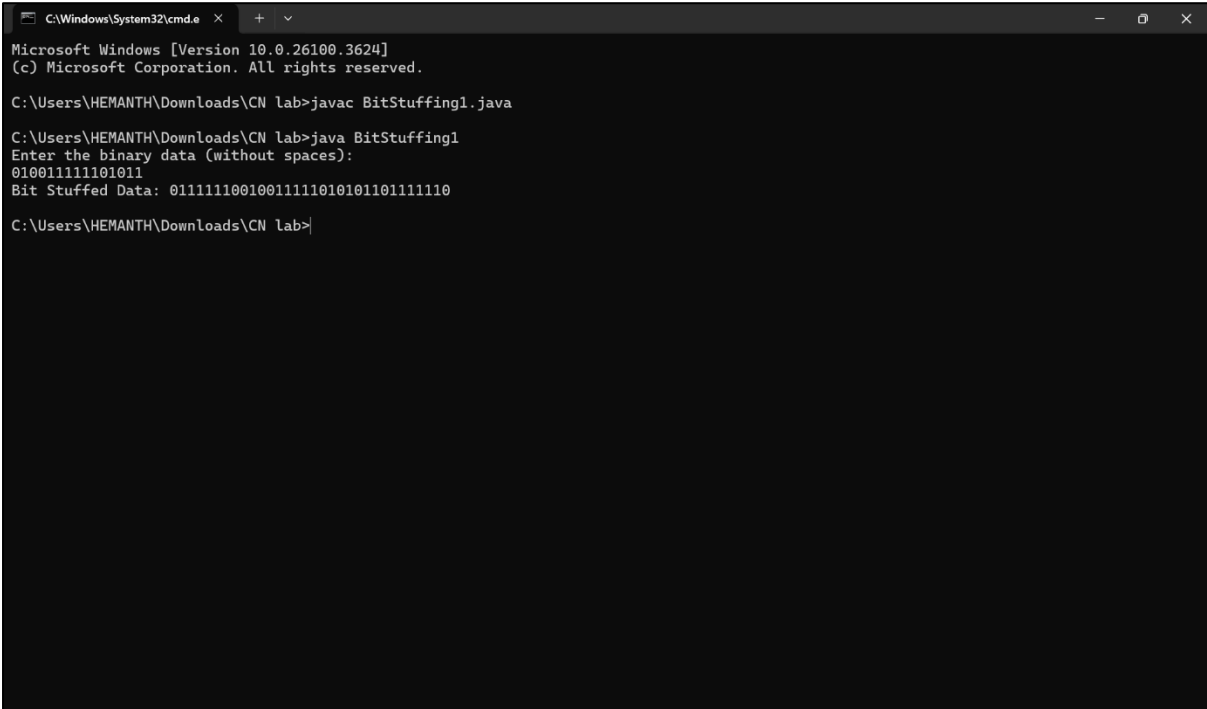
```
import java.util.Scanner;

public class BitStuffing1 {

    public static String bitStuffing(String inputData) {
        StringBuilder stuffedData = new StringBuilder();
        int count = 0;
        stuffedData.append("01111110");
        for (int i = 0; i < inputData.length(); i++) {
            stuffedData.append(inputData.charAt(i));
            if (inputData.charAt(i) == '1') {
                count++;
                if (count == 5) {
                    stuffedData.append('0');
                    count = 0;
                }
            } else {
                count = 0;
            }
        }
        stuffedData.append("01111110");
        return stuffedData.toString();
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the binary data (without spaces):");
        String inputData = scanner.nextLine();
        String stuffedData = bitStuffing(inputData);
        System.out.println("Bit Stuffed Data: " + stuffedData);
    }
}
```

Output:



```
C:\Windows\System32\cmd.e x + v
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH\Downloads\CN lab>javac BitStuffing1.java

C:\Users\HEMANTH\Downloads\CN lab>java BitStuffing1
Enter the binary data (without spaces):
010011111101011
Bit Stuffed Data: 0111111001001111101010110111110

C:\Users\HEMANTH\Downloads\CN lab>
```

Description: Bit destuffing is the process used by the receiver to remove the extra bits added during bit stuffing. After detecting five consecutive '1's in the received data, the receiver removes the '0' that follows. This restores the original data before it was transmitted.

BitDestuffing:

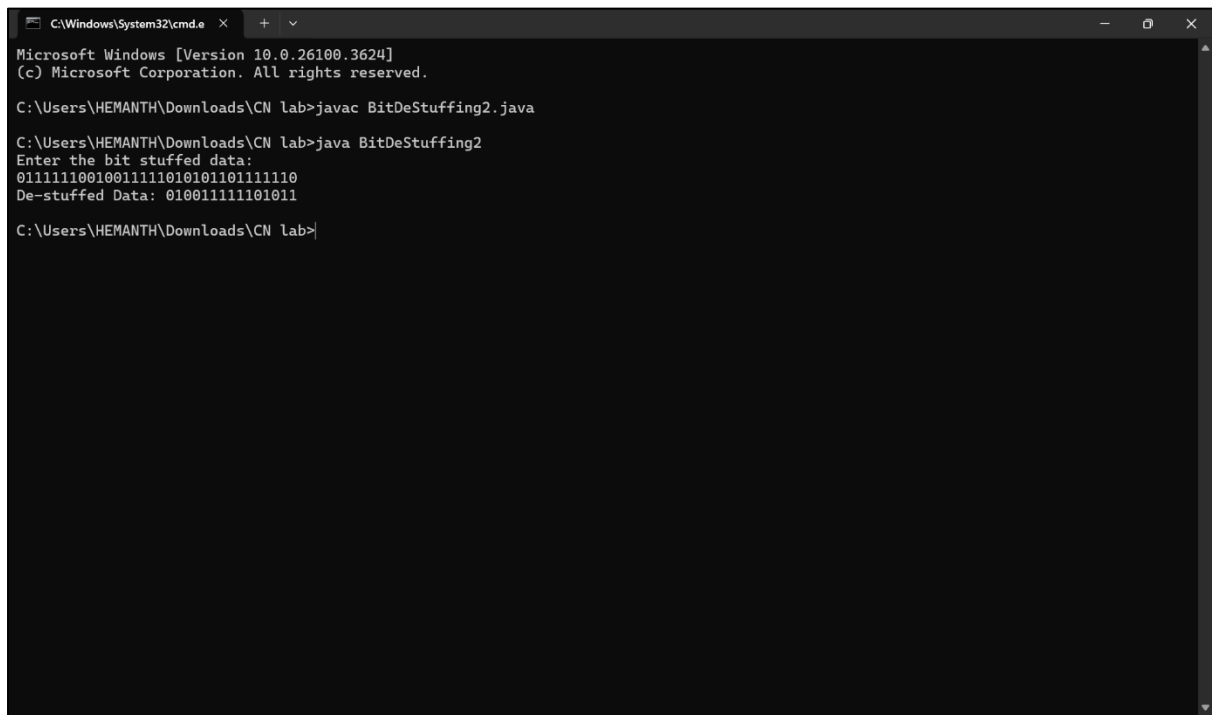
```
import java.util.Scanner;

public class BitDeStuffing2 {

    public static String bitDeStuffing(String stuffedData) {
        StringBuilder deStuffedData = new StringBuilder();
        int count = 0;
        String dataWithoutFlags = stuffedData.substring(8, stuffedData.length() - 8);
        for (int i = 0; i < dataWithoutFlags.length(); i++) {
            deStuffedData.append(dataWithoutFlags.charAt(i));
            if (dataWithoutFlags.charAt(i) == '1') {
                count++;
                if (count == 5 && i + 1 < dataWithoutFlags.length() && dataWithoutFlags.charAt(i + 1) ==
'0') {
                    i++;
                    count = 0;
                }
            } else {
                count = 0;
            }
        }
        return deStuffedData.toString();
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the bit stuffed data:");
        String stuffedData = scanner.nextLine();
        String deStuffedData = bitDeStuffing(stuffedData);
        System.out.println("De-stuffed Data: " + deStuffedData);
    }
}
```

Output:



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH\Downloads\CN lab>javac BitDeStuffing2.java

C:\Users\HEMANTH\Downloads\CN lab>java BitDeStuffing2
Enter the bit stuffed data:
01111110010011111010101101111110
De-stuffed Data: 010011111101011

C:\Users\HEMANTH\Downloads\CN lab>
```


Experiment-4

AIM: To implement byte stuffing (or) character stuffing.

Description: Byte stuffing is a technique used to avoid confusion with special control bytes (like frame start or end). When a special byte appears in the data, an **escape byte** is added before it. The receiver removes the escape byte during decoding to get the original data.

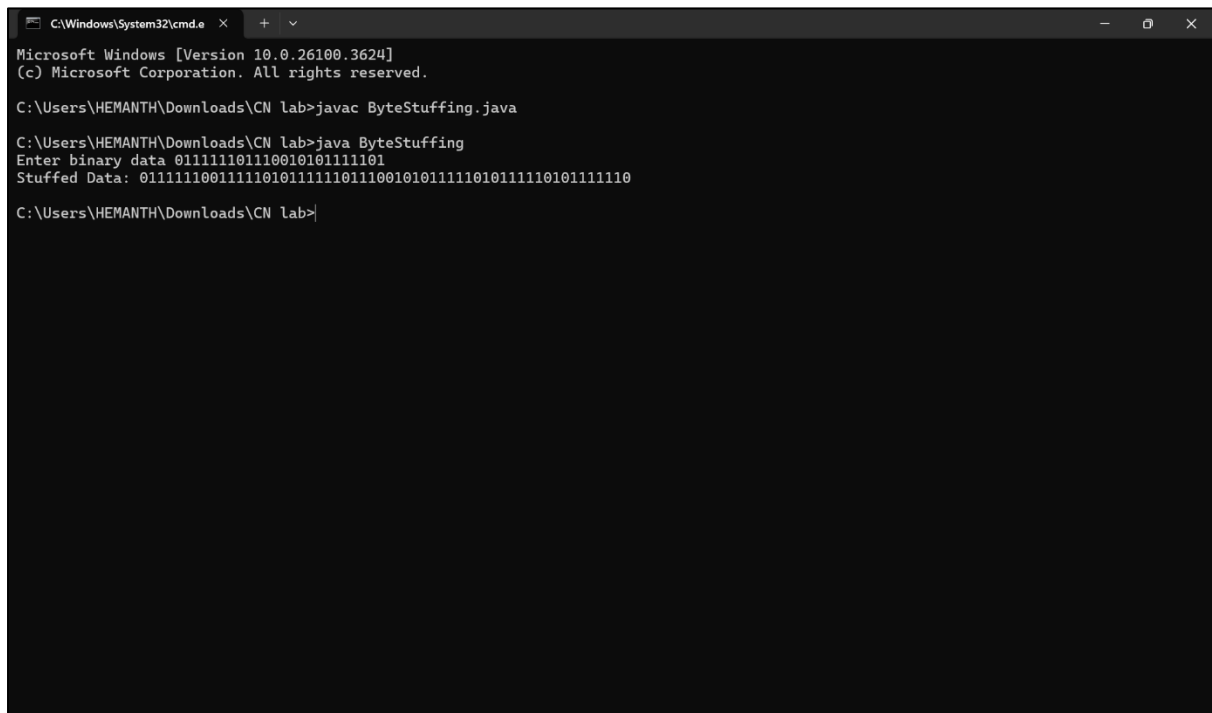
ByteStuffing:

```
import java.util.Scanner;

public class ByteStuffing {
    private static final String FLAG = "01111110";
    private static final String ESCAPE = "01111101";
    public static String bitStuffing(String data) {
        StringBuilder stuffedData = new StringBuilder(FLAG);
        for (int i = 0; i < data.length(); i += 8) {
            String byteChunk = data.substring(i, Math.min(i + 8, data.length()));
            if (byteChunk.equals(FLAG) || byteChunk.equals(ESCAPE)) {
                stuffedData.append(ESCAPE);
            }
            stuffedData.append(byteChunk);
        }
        stuffedData.append(FLAG);
        return stuffedData.toString();
    }
    public static boolean isValidBinary(String data) {
        return data.matches("[01]+");
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter binary data ");
        String input = scanner.nextLine();
        if (!isValidBinary(input)) {
            System.out.println("Invalid input! Please enter only binary digits (0's and 1's).");
        } else {
            String stuffedData = bitStuffing(input);
            System.out.println("Stuffed Data: " + stuffedData);
        }
        scanner.close();
    }
}
```

```
}  
}
```

Output:



```
C:\Windows\System32\cmd.exe x + v  
Microsoft Windows [Version 10.0.26100.3624]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\HEMANTH\Downloads\CN lab>javac ByteStuffing.java  
  
C:\Users\HEMANTH\Downloads\CN lab>java ByteStuffing  
Enter binary data 011111101110010101111101  
Stuffed Data: 01111110011111101011111011111010101111101011111010111110  
  
C:\Users\HEMANTH\Downloads\CN lab>
```

Description: Byte destuffing is the process used by the receiver to remove the escape bytes added during byte stuffing. When an escape byte is detected, the next byte is treated as data—even if it's a control character—restoring the original message.

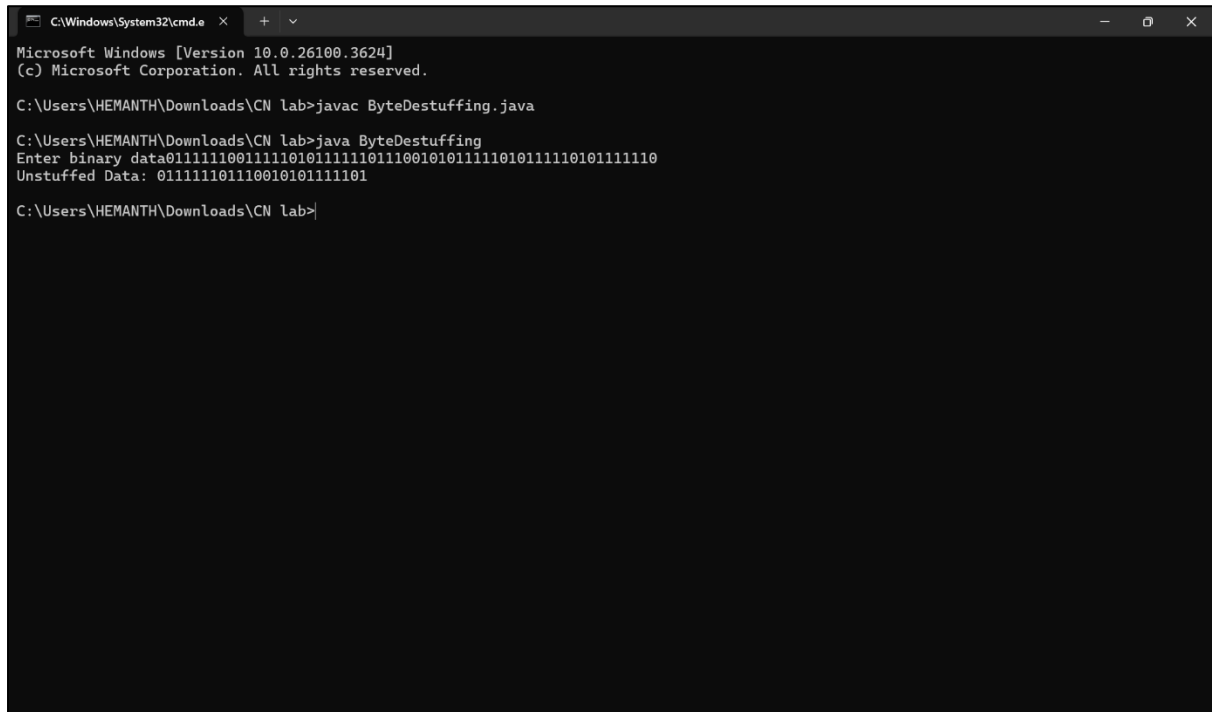
ByteDeStuffing:

```
import java.util.Scanner;

public class ByteDestuffing {
    private static final String FLAG = "01111110";
    private static final String ESCAPE = "01111101";
    public static String bitUnstuffing(String stuffedData) {
        String data = stuffedData.substring(FLAG.length(), stuffedData.length() - FLAG.length());
        StringBuilder unstuffedData = new StringBuilder();
        int i = 0;
        while (i < data.length()) {
            if (i + ESCAPE.length() <= data.length() && data.substring(i, i +
ESCAPE.length()).equals(ESCAPE)) {
                i += ESCAPE.length();
                if (i + 8 <= data.length()) {
                    unstuffedData.append(data.substring(i, i + 8));
                    i += 8;
                }
            } else {
                unstuffedData.append(data.substring(i, i + 8));
                i += 8;
            }
        }
        return unstuffedData.toString();
    }
    public static boolean isValidBinary(String data) {
        return data.matches("[01]+");
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter binary data");
        String input = scanner.nextLine();
        if (!isValidBinary(input)) {
            System.out.println("Invalid input! Please enter only binary digits (0's and 1's).");
        } else {
```

```
String unstuffedData = bitUnstuffing(input);  
System.out.println("Unstuffed Data: " + unstuffedData);  
}  
scanner.close();  
}  
}
```

Output:



```
C:\Windows\System32\cmd.exe X + v  
Microsoft Windows [Version 10.0.26100.3624]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\HEMANTH\Downloads\CN lab>javac ByteDestuffing.java  
  
C:\Users\HEMANTH\Downloads\CN lab>java ByteDestuffing  
Enter binary data011111100111110101111110111110010101111101011111010111110  
Unstuffed Data: 011111101110010101111101  
  
C:\Users\HEMANTH\Downloads\CN lab>
```

Experiment-5

Aim: To implement stop and wait protocol program.

Description: The Stop and Wait protocol is a simple data link layer method for reliable communication. The sender sends one frame at a time and waits for an acknowledgment (ACK) before sending the next frame. If no ACK is received, the sender retransmits the frame. This ensures error-free and orderly delivery, but is slower due to waiting after each frame.

Client:

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

public class StopAndWaitClient {
    public static void main(String[] args) throws Exception {
        DatagramSocket clientSocket = new DatagramSocket();
        clientSocket.setSoTimeout(2000);
        InetAddress serverAddress = InetAddress.getByName("localhost");
        Scanner userInput = new Scanner(System.in);
        byte[] sendBuffer;
        byte[] receiveBuffer = new byte[1024];
        System.out.print("Enter number of frames to send: ");
        int totalFrames = userInput.nextInt();
        userInput.nextLine();
        for (int frameIndex = 0; frameIndex < totalFrames; ) {
            System.out.print("Enter frame number: ");
            String frameNumber = userInput.nextLine();
            System.out.print("Enter message for frame " + frameNumber + ": ");
            String message = userInput.nextLine();
            String fullFrame = "Frame " + frameNumber + ": " + message;
            sendBuffer = fullFrame.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendBuffer,
                sendBuffer.length, serverAddress, 9876);
            DatagramPacket ackPacket = new DatagramPacket(receiveBuffer,
                receiveBuffer.length);
            boolean isAckReceived = false;
            while (!isAckReceived) {
                clientSocket.send(sendPacket);
                System.out.println("Sent --> " + fullFrame);
```

```
        try {
            clientSocket.receive(ackPacket);
            String ackMessage = new String(ackPacket.getData(), 0,
ackPacket.getLength());
            System.out.println("ACK received --> " + ackMessage);
            isAckReceived = true;
            frameIndex++; // Move to next frame
        } catch (SocketTimeoutException e) {
            System.out.println("Timeout! No ACK for frame " + frameNumber +
". Resending...");
        }
    }
}
clientSocket.close();
}
```

Server:

```
import java.net.*;
import java.util.Random;
public class StopAndWaitServer {
    public static void main(String[] args) throws Exception {
        DatagramSocket serverSocket = new DatagramSocket(9876);
        byte[] buffer = new byte[1024];
        Random random = new Random();
        System.out.println("Server is running and waiting for frames...");
        while (true) {
            DatagramPacket incomingPacket = new DatagramPacket(buffer,
buffer.length);
            serverSocket.receive(incomingPacket);
            String receivedFrame = new String(incomingPacket.getData(), 0,
incomingPacket.getLength());
            System.out.println("Received: " + receivedFrame);
            boolean shouldSendAck = random.nextBoolean();
            if (shouldSendAck) {
                String ackMessage = "ACK";
                byte[] ackBuffer = ackMessage.getBytes();
                DatagramPacket ackPacket = new DatagramPacket(
                    ackBuffer,
                    ackBuffer.length,
                    incomingPacket.getAddress(),
                    incomingPacket.getPort()
                );
                serverSocket.send(ackPacket);
                System.out.println("ACK sent for: " + receivedFrame);
            } else {
                System.out.println("Simulated ACK loss for: " + receivedFrame);
            }
        }
    }
}
```

Output:

```

C:\Windows\System32\cmd.e x + v - □ x
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH\Downloads\new hemanth file>javac StopAndWaitServer.java

C:\Users\HEMANTH\Downloads\new hemanth file>java StopAndWaitServer
Server is running and waiting for frames...
Received: Frame 1: cn
ACK sent for: Frame 1: cn
Received: Frame 2: lab
Simulated ACK loss for: Frame 2: lab
Received: Frame 2: lab
Simulated ACK loss for: Frame 2: lab
Received: Frame 2: lab
ACK sent for: Frame 2: lab
Received: Frame 3: tcp
Simulated ACK loss for: Frame 3: tcp
Received: Frame 3: tcp
ACK sent for: Frame 3: tcp
Received: Frame 4: udp
ACK sent for: Frame 4: udp
|

C:\Windows\System32\cmd.e x + v - □ x
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH\Downloads\new hemanth file>javac StopAndWaitClient.java

C:\Users\HEMANTH\Downloads\new hemanth file>java StopAndWaitClient
Enter number of frames to send: 4
Enter frame number: 1
Enter message for frame 1: cn
Sent --> Frame 1: cn
ACK received --> ACK
Enter frame number: 2
Enter message for frame 2: lab
Sent --> Frame 2: lab
Timeout! No ACK for frame 2. Resending...
Sent --> Frame 2: lab
Timeout! No ACK for frame 2. Resending...
Sent --> Frame 2: lab
ACK received --> ACK
Enter frame number: 3
Enter message for frame 3: tcp
Sent --> Frame 3: tcp
Timeout! No ACK for frame 3. Resending...
Sent --> Frame 3: tcp
ACK received --> ACK
Enter frame number: 4
Enter message for frame 4: udp
Sent --> Frame 4: udp
ACK received --> ACK

C:\Users\HEMANTH\Downloads\new hemanth file>|

```


Experiment-6

AIM: To implement the dijkstra's algorithm.

Description: Dijkstra's Algorithm is used to find the **shortest path** from a source node to all other nodes in a weighted graph with non-negative edges. It works by selecting the nearest unvisited node, updating the distances to its neighbors, and repeating the process until all nodes are visited.

Code:

```
import java.util.*;

public class DijkstraAlgorithm2 {
    private static final int INF = 999;

    public static void dijkstra(int[][] graph, int src) {
        int vertices = graph.length;
        int[] dist = new int[vertices];
        boolean[] visited = new boolean[vertices];
        Arrays.fill(dist, INF);
        dist[src] = 0;
        for (int count = 0; count < vertices - 1; count++) {
            int u = minDistance(dist, visited, vertices);
            visited[u] = true;
            for (int v = 0; v < vertices; v++) {
                if (!visited[v] && graph[u][v] != INF && dist[u] != INF && dist[u] + graph[u][v] < dist[v])
                {
                    dist[v] = dist[u] + graph[u][v];
                }
            }
        }
        printSolution(dist);
    }

    private static int minDistance(int[] dist, boolean[] visited, int vertices) {
        int min = INF, minIndex = -1;
        for (int v = 0; v < vertices; v++) {
            if (!visited[v] && dist[v] < min) {
                min = dist[v];
                minIndex = v;
            }
        }
        return minIndex;
    }
}
```

```
}  
private static void printSolution(int[] dist) {  
    System.out.println("Vertex      Distance from Source");  
    for (int i = 0; i < dist.length; i++) {  
        System.out.println(i + "    " + (dist[i] == INF ? "INF" : dist[i]));  
    }  
}  
public static void main(String[] args) {  
    Scanner scanner = new Scanner(System.in);  
    System.out.print("Enter the number of vertices: ");  
    int vertices = scanner.nextInt();  
    int[][] graph = new int[vertices][vertices];  
  
    System.out.println("Enter the adjacency matrix");  
    for (int i = 0; i < vertices; i++) {  
        for (int j = 0; j < vertices; j++) {  
            graph[i][j] = scanner.nextInt();  
        }  
    }  
    System.out.print("Enter the source vertex: ");  
    int src = scanner.nextInt();  
    dijkstra(graph, src);  
    scanner.close();  
}  
}
```

Output:

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH\Downloads\CN lab>javac DijkstraAlgorithm2.java

C:\Users\HEMANTH\Downloads\CN lab>java DijkstraAlgorithm2
Enter the number of vertices: 7
Enter the adjacency matrix
0 2 5 999 999 999 999
2 0 999 3 999 999 999
5 999 0 4 999 999 999
999 3 4 0 7 1 999
999 999 999 7 0 999 3
999 999 999 1 999 0 5
999 999 999 999 3 5 0
Enter the source vertex: 0
Vertex Distance from Source
0 0
1 2
2 5
3 5
4 12
5 6
6 11

C:\Users\HEMANTH\Downloads\CN lab>
```


Experiment-7

AIM: To implement a DistanceVectorRouting program.

Description: Distance Vector Routing is a routing protocol where each router shares its routing table with its neighbors. Routers update their tables based on the shortest path (least cost) to each destination, using the Bellman-Ford algorithm. Updates happen periodically or when there are changes in the network.

Code:

```
import java.util.*;

class DistanceVectorRouting {

    public static void main(String[] args) {

        int totalNodes;

        Scanner inputScanner = new Scanner(System.in);

        System.out.println("Enter the number of nodes: ");

        totalNodes = inputScanner.nextInt();

        int distanceMatrix[][] = new int[totalNodes][totalNodes];

        System.out.println("Enter the distance matrix: ");

        for (int x = 0; x < totalNodes; x++) {

            for (int y = 0; y < totalNodes; y++) {

                distanceMatrix[x][y] = inputScanner.nextInt();

            }

        }

        int[][] routingData = new int[totalNodes][totalNodes];

        int[][] nextHop = new int[totalNodes][totalNodes];

        for (int x = 0; x < totalNodes; x++) {

            for (int y = 0; y < totalNodes; y++) {

                if (distanceMatrix[x][y] == 0) {

                    if (x == y) {

                        routingData[x][y] = 0;

                    } else {

                        routingData[x][y] = 999;

                    }

                    nextHop[x][y] = -1;

                } else {

                    routingData[x][y] = distanceMatrix[x][y]; // Direct link cost

                    nextHop[x][y] = y; // Direct link, next hop is destination itself

                }

            }

        }

    }

}
```

```

    }
}
boolean tableUpdated;
do {
    tableUpdated = false;
    for (int x = 0; x < totalNodes; x++) {
        for (int y = 0; y < totalNodes; y++) {
            if (x != y) { // Ignore the case where the source and destination are the same
                for (int z = 0; z < totalNodes; z++) {
                    if (routingData[x][z] + routingData[z][y] < routingData[x][y]) {
                        routingData[x][y] = routingData[x][z] + routingData[z][y];
                        nextHop[x][y] = nextHop[x][z];
                        tableUpdated = true;
                    }
                }
            }
        }
    }
} while (tableUpdated);
System.out.println("Routing table with destination, cost, and next hop: ");
for (int x = 0; x < totalNodes; x++) {
    System.out.println("Router " + x + "'s Routing Table:");
    for (int y = 0; y < totalNodes; y++) {
        if (routingData[x][y] == 999) {
            System.out.println("Destination: " + y + " | Cost: 999 | Next Hop: None");
        } else {
            System.out.println("Destination: " + y + " | Cost: " + routingData[x][y] + " | Next Hop: " +
nextHop[x][y]);
        }
    }
    System.out.println();
}
inputScanner.close();
}
}

```

Output:

```

C:\Windows\System32\cmd.e X + v

C:\Users\HEMANTH\Downloads\CN Lab>javac DistanceVectorRouting.java

C:\Users\HEMANTH\Downloads\CN Lab>java DistanceVectorRouting
Enter the number of nodes:
5
Enter the distance matrix:
0 2 4 999 2
2 0 999 5 999
4 999 0 2 999
999 5 2 0 3
2 999 999 3 0
Routing table with destination, cost, and next hop:
Router 0's Routing Table:
Destination: 0 | Cost: 0 | Next Hop: -1
Destination: 1 | Cost: 2 | Next Hop: 1
Destination: 2 | Cost: 4 | Next Hop: 2
Destination: 3 | Cost: 5 | Next Hop: 4
Destination: 4 | Cost: 2 | Next Hop: 4

Router 1's Routing Table:
Destination: 0 | Cost: 2 | Next Hop: 0
Destination: 1 | Cost: 0 | Next Hop: -1
Destination: 2 | Cost: 6 | Next Hop: 0
Destination: 3 | Cost: 5 | Next Hop: 3
Destination: 4 | Cost: 4 | Next Hop: 0

Router 2's Routing Table:
Destination: 0 | Cost: 4 | Next Hop: 0
Destination: 1 | Cost: 6 | Next Hop: 0
Destination: 2 | Cost: 0 | Next Hop: -1
Destination: 3 | Cost: 2 | Next Hop: 3
Destination: 4 | Cost: 5 | Next Hop: 3

Router 3's Routing Table:
Destination: 0 | Cost: 5 | Next Hop: 4
Destination: 1 | Cost: 5 | Next Hop: 1
Destination: 2 | Cost: 2 | Next Hop: 2
Destination: 3 | Cost: 0 | Next Hop: -1
Destination: 4 | Cost: 3 | Next Hop: 4

Router 4's Routing Table:
Destination: 0 | Cost: 2 | Next Hop: 0
Destination: 1 | Cost: 4 | Next Hop: 0
Destination: 2 | Cost: 5 | Next Hop: 3
Destination: 3 | Cost: 3 | Next Hop: 3
Destination: 4 | Cost: 0 | Next Hop: -1

C:\Users\HEMANTH\Downloads\CN Lab>

```


Experiment-8

AIM: To implement a program on leakybucket algorithm.

Description: The Leaky Bucket Algorithm is used for **traffic shaping** and **rate control** in networks. Data packets are added to a bucket (buffer) and sent out at a **fixed rate**, like water leaking from a bucket. If the bucket overflows (too much data too fast), excess packets are discarded, helping prevent congestion.

Code:

```
import java.util.Scanner;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
class LeakyBucketSystem {
    private final double capacity;
    private double currentWater;
    private final double fillRate;
    private final double leakRate;
    private boolean running = true;
    public LeakyBucketSystem(double capacity, double fillRate, double leakRate) {
        this.capacity = capacity;
        this.fillRate = fillRate;
        this.leakRate = leakRate;
        this.currentWater = 0;
    }
    public void startSimulation(int duration) {
        Thread fillThread = new Thread(this::fill);
        Thread leakThread = new Thread(this::leak);
        Thread monitorThread = new Thread(() -> monitor(duration));
        fillThread.start();
        leakThread.start();
        monitorThread.start();
        try {
            monitorThread.join();
            running = false;
            fillThread.join();
            leakThread.join();
        } catch (InterruptedException e) {
            System.out.println("Simulation interrupted.");
        }
    }
}
```

```

    }
}
private void fill() {
    while (running) {
        synchronized (this) {
            currentWater += fillRate;
            if (currentWater > capacity) {
                System.out.println("[Overflow] Wasting " + (currentWater - capacity) + " Bytes.");
                currentWater = capacity;
            }
        }
        sleep(1000);
    }
}
private void leak() {
    while (running) {
        synchronized (this) {
            currentWater -= leakRate;
            if (currentWater < 0) {
                currentWater = 0;
            }
        }
        sleep(1000);
    }
}
private void monitor(int duration) {
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");
    for (int i = 0; i < duration; i++) {
        synchronized (this) {
            String formattedTime = LocalDateTime.now().format(formatter);
            System.out.println "[" + formattedTime + "] Time: " + (i + 1) + "s | Water Level: " +
currentWater + "Bytes");
        }
        sleep(1000);
    }
}
private void sleep(int millis) {

```

```
    try {
        Thread.sleep(millis);
    } catch (InterruptedException e) {
        System.out.println("Thread interrupted.");
    }
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter bucket capacity: ");
    double capacity = scanner.nextDouble();
    System.out.print("Enter water inflow rate: ");
    double fillRate = scanner.nextDouble();
    System.out.print("Enter water leak rate: ");
    double leakRate = scanner.nextDouble();
    LeakyBucketSystem bucket = new LeakyBucketSystem(capacity, fillRate, leakRate);
    while (true) {
        System.out.print("Enter simulation duration (seconds): ");
        int duration = scanner.nextInt();
        bucket.startSimulation(duration);
        System.out.print("Do you want to continue the simulation? (yes/no): ");
        String choice = scanner.next();
        if (!choice.equalsIgnoreCase("yes")) {
            break;
        }
    }
    scanner.close();
}
```

Output:

```
C:\Windows\System32\cmd.e X + v
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH\Downloads\CN lab>javac LeakyBucketSystem.java

C:\Users\HEMANTH\Downloads\CN lab>java LeakyBucketSystem
Enter bucket capacity: 12
Enter water inflow rate: 5
Enter water leak rate: 2
Enter simulation duration (seconds): 7
[2025-04-06 14:45:05] Time: 1s | Water Level: 3.0Bytes
[2025-04-06 14:45:06] Time: 2s | Water Level: 6.0Bytes
[2025-04-06 14:45:07] Time: 3s | Water Level: 9.0Bytes
[2025-04-06 14:45:08] Time: 4s | Water Level: 12.0Bytes
[Overflow] Wasting 3.0 Bytes.
[2025-04-06 14:45:09] Time: 5s | Water Level: 12.0Bytes
[Overflow] Wasting 3.0 Bytes.
[2025-04-06 14:45:10] Time: 6s | Water Level: 12.0Bytes
[Overflow] Wasting 3.0 Bytes.
[2025-04-06 14:45:11] Time: 7s | Water Level: 12.0Bytes
[Overflow] Wasting 3.0 Bytes.
Do you want to continue the simulation? (yes/no): |
```

Experiment-9

AIM: To implement Tcp client and server program.

Description: In a TCP connection, the **server** listens for incoming requests on a specific port, while the **client** initiates the connection. TCP (Transmission Control Protocol) ensures reliable, ordered, and error-free communication between the two. Once connected, they can exchange data until the connection is closed.

Client:

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

class TCPClient {
    public static void main(String[] args) {
        String serverHost = "localhost";
        int serverPort = 7000;
        try (Socket clientSocket = new Socket(serverHost, serverPort);
            BufferedReader serverInput = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
            PrintWriter serverOutput = new PrintWriter(clientSocket.getOutputStream(), true);
            BufferedReader userConsoleInput = new BufferedReader(new
InputStreamReader(System.in))) {
            System.out.println(serverInput.readLine());
            System.out.println("Connected to server. Type messages (type 'exit' to quit):");
            String userMessage;
            while (true) {
                System.out.print("You: ");
                userMessage = userConsoleInput.readLine();
                serverOutput.println(userMessage);

                if ("exit".equalsIgnoreCase(userMessage)) {
                    break;
                }
                String serverResponse = serverInput.readLine();
                System.out.println(serverResponse);
            }
            System.out.println("Connection closed.");
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

```
}  
}  
}
```

Server:

```

import java.io.*;
import java.net.*;
class TCPServer {
    private static final int SERVER_PORT = 7000;
    private static int clientCount = 0;
    public static void main(String[] args) {
        try (ServerSocket serverListener = new ServerSocket(SERVER_PORT)) {
            System.out.println("Server is running. Awaiting for connections..");
            while (true) {
                Socket clientSocket = serverListener.accept();
                clientCount++;
                System.out.println("Client " + clientCount + " connected.");
                new ClientHandlerThread(clientSocket, clientCount).start();
            }
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}

class ClientHandlerThread extends Thread {
    private Socket clientSocket;
    private int clientID;

    public ClientHandlerThread(Socket clientSocket, int clientID) {
        this.clientSocket = clientSocket;
        this.clientID = clientID;
    }

    public void run() {
        try (
            BufferedReader reader = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
            PrintWriter writer = new PrintWriter(clientSocket.getOutputStream(), true)
        ) {
            writer.println("Welcome, Client " + clientID + " ");
            String clientInput;
            while ((clientInput = reader.readLine()) != null) {

```

```

        System.out.println("Client " + clientID + ": " + clientInput);
        writer.println("Server received: " + clientInput);
        if (clientInput.equalsIgnoreCase("exit")) {
            break;
        }
    }
    clientSocket.close();
    System.out.println("Client " + clientID + " disconnected.");
} catch (IOException ex) {
    ex.printStackTrace();
}
}
}

```

Output:

```

C:\Windows\System32\cmd.e x + v
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH\Downloads\tcp new>javac TCPServer.java

C:\Users\HEMANTH\Downloads\tcp new>java TCPServer
Server is running. Awaiting for connections..
Client 1 connected.
Client 2 connected.
Client 1: hi
Client 1: hello
Client 2: how are you
Client 2: fine
Client 1: exit
Client 1 disconnected.
Client 2: exit
Client 2 disconnected.

C:\Windows\System32\cmd.e x + v
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH\Downloads\tcp new>javac TCPClient.java

C:\Users\HEMANTH\Downloads\tcp new>java TCPClient
Welcome, Client 1
Connected to server. Type messages (type 'exit' to quit):
You: hi
Server received: hi
You: hello
Server received: hello
You: exit
Connection closed.

C:\Users\HEMANTH\Downloads\tcp new>

C:\Windows\System32\cmd.e x + v
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH\Downloads\tcp new>javac TCPClient.java

C:\Users\HEMANTH\Downloads\tcp new>java TCPClient
Welcome, Client 2
Connected to server. Type messages (type 'exit' to quit):
You: how are you
Server received: how are you
You: fine
Server received: fine
You: exit
Connection closed.

C:\Users\HEMANTH\Downloads\tcp new>

```


Experiment-10

AIM: To implement udp client and server program.

Description: In a UDP (User Datagram Protocol) connection, the server listens on a port and the client sends messages (datagrams) to it. Unlike TCP, UDP is connectionless, faster, and does not guarantee delivery, order, or error checking—making it suitable for real-time applications like video streaming or gaming.

Client:

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

public class UDPClient {
    private static final String HOST = "localhost";
    private static final int PORT = 8080;

    public static void main(String[] args) {
        try (DatagramSocket udpSocket = new DatagramSocket();
            BufferedReader consoleInput = new BufferedReader(new InputStreamReader(System.in))) {
            InetAddress serverHost = InetAddress.getByName(HOST);
            byte[] receiveBuffer = new byte[1024];
            System.out.println("Connected to server");
            while (true) {
                System.out.print("You: ");
                String userMessage = consoleInput.readLine();
                DatagramPacket sendPacket = new DatagramPacket(userMessage.getBytes(),
userMessage.length(), serverHost, PORT);
                udpSocket.send(sendPacket);
                if ("exit".equalsIgnoreCase(userMessage)) {
                    System.out.println("Client disconnected.");
                    break;
                }
                DatagramPacket responsePacket = new DatagramPacket(receiveBuffer,
receiveBuffer.length);
                udpSocket.receive(responsePacket);
                System.out.println("Server:" + new String(responsePacket.getData(), 0,
responsePacket.getLength()));
            }
        } catch (IOException ex) {
            System.out.println("Connection error: " + ex.getMessage());
        }
    }
}
```

```
}  
}  
}
```

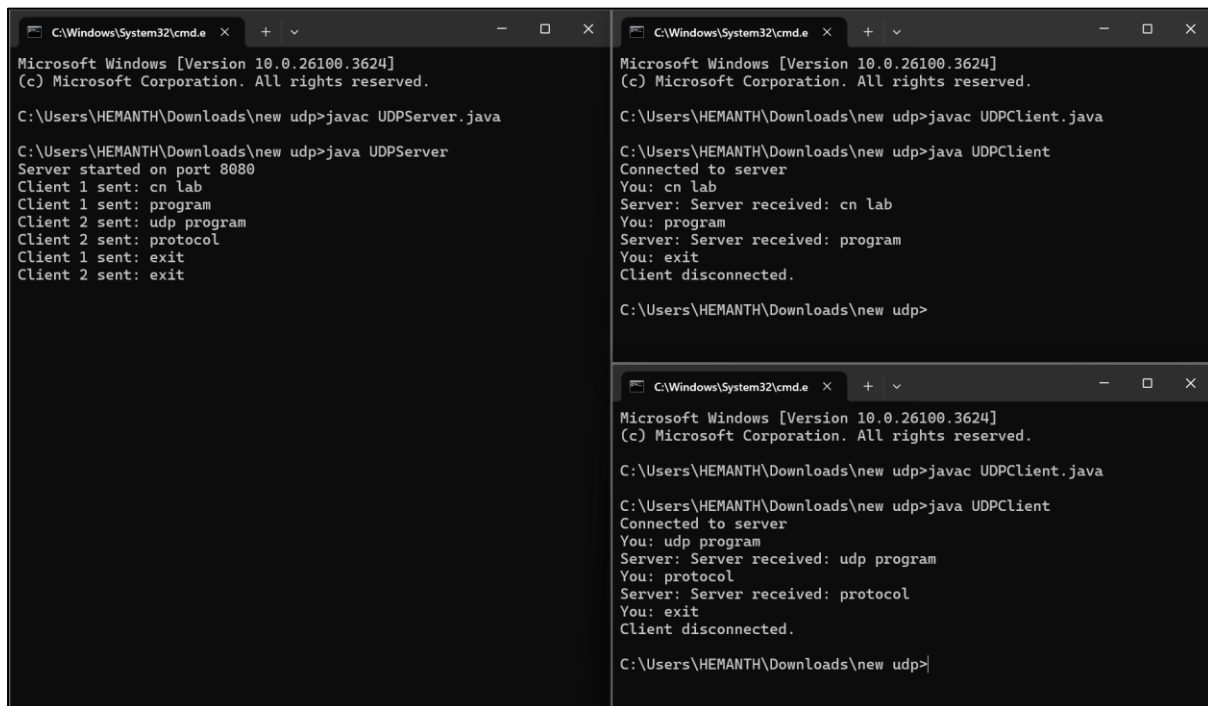
Server:

```

import java.io.*;
import java.net.*;
import java.util.concurrent.ConcurrentHashMap;
public class UDPServer {
    private static final int SERVER_PORT = 8080;
    private static boolean isRunning = true;
    private static final ConcurrentHashMap<String, Integer> connectedClients = new
ConcurrentHashMap<>();
    public static void main(String[] args) {
        try (DatagramSocket udpSocket = new DatagramSocket(SERVER_PORT)) {
            System.out.println("Server started on port " + SERVER_PORT);
            byte[] receiveBuffer = new byte[1024];
            while (isRunning) {
                DatagramPacket receivedPacket = new DatagramPacket(receiveBuffer,
receiveBuffer.length);
                udpSocket.receive(receivedPacket);
                String clientKey = receivedPacket.getAddress() + ":" + receivedPacket.getPort();
                connectedClients.putIfAbsent(clientKey, connectedClients.size() + 1);
                int clientId = connectedClients.get(clientKey);
                String clientMessage = new String(receivedPacket.getData(), 0, receivedPacket.getLength());
                System.out.println("Client " + clientId + " sent: " + clientMessage);
                if ("exit".equalsIgnoreCase(clientMessage)) {
                    connectedClients.remove(clientKey);
                    continue;
                }
                byte[] responseBuffer = ("Server received: " + clientMessage).getBytes();
                DatagramPacket responsePacket = new DatagramPacket(responseBuffer,
responseBuffer.length,
                    receivedPacket.getAddress(), receivedPacket.getPort());
                udpSocket.send(responsePacket);
            }
        } catch (IOException ex) {
            System.out.println("Server error: " + ex.getMessage());
        }
    }
}

```

Output:



The image displays three separate Windows command prompt windows, each showing the execution of a Java program. The first window on the left shows the compilation of `UDPServer.java` and its execution. The server starts on port 8080 and receives two client messages: "cn lab" and "program", followed by "udp program" and "protocol" from Client 2. Both clients then send "exit". The top-right window shows the compilation of `UDPClient.java` and its execution. The client connects to the server, sends "cn lab" and "program", receives "Server received: cn lab" and "Server received: program" from the server, and then sends "exit" before disconnecting. The bottom-right window shows another execution of `UDPClient.java`, where the client sends "udp program" and "protocol", receives "Server received: udp program" and "Server received: protocol" from the server, and then sends "exit" before disconnecting.

```
C:\Windows\System32\cmd.e x + v - □ x
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH\Downloads\new udp>javac UDPServer.java

C:\Users\HEMANTH\Downloads\new udp>java UDPServer
Server started on port 8080
Client 1 sent: cn lab
Client 1 sent: program
Client 2 sent: udp program
Client 2 sent: protocol
Client 1 sent: exit
Client 2 sent: exit

C:\Windows\System32\cmd.e x + v - □ x
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH\Downloads\new udp>javac UDPClient.java

C:\Users\HEMANTH\Downloads\new udp>java UDPClient
Connected to server
You: cn lab
Server: Server received: cn lab
You: program
Server: Server received: program
You: exit
Client disconnected.

C:\Users\HEMANTH\Downloads\new udp>

C:\Windows\System32\cmd.e x + v - □ x
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HEMANTH\Downloads\new udp>javac UDPClient.java

C:\Users\HEMANTH\Downloads\new udp>java UDPClient
Connected to server
You: udp program
Server: Server received: udp program
You: protocol
Server: Server received: protocol
You: exit
Client disconnected.

C:\Users\HEMANTH\Downloads\new udp>
```