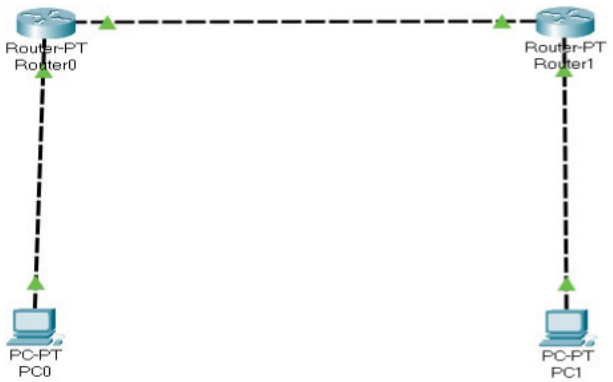
Practical No. 1
Study of Networking

Aim :- To study the IP address.

Circuit Diagram :-



Program :-
Router 0 > cli :- enable configure terminal interface fastethernet 0/0 ip address 10.0.0.1 255.0.0.0 no shutdown

exit interface fastethernet 0/1 ip address 20.0.0.1 255.0.0.0 no shutdown

exit

Router 1 > cli :- enable configure terminal interface fastethernet 0/0 ip address 40.0.0.1 255.0.0.0 no shutdown

exit

interface fastethernet 0/1 ip address 20.0.0.2 255.0.0.0 no shutdown

exit

Pc 0 > Desktop > Ip configuration :- ipv4 :
10.0.0.2 subnet mask : 255.0.0.0

Default gateway: 10.0.0.1

Pc 1 > Desktop > Ip configuration :-

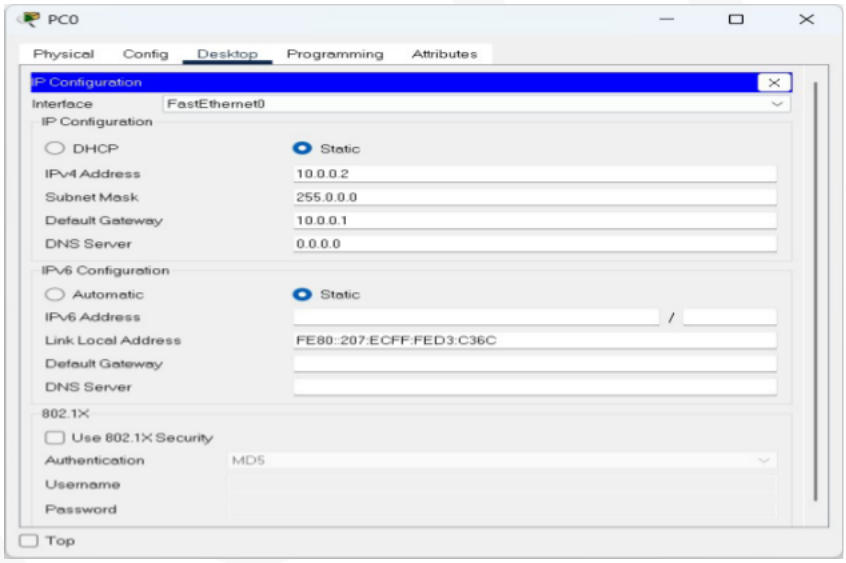
ipv4 : 40.0.0.2

subnet mask : 255.0.0.0

Default gateway: 40.0.0.1

Output :-

PC0 :-



PC1:-

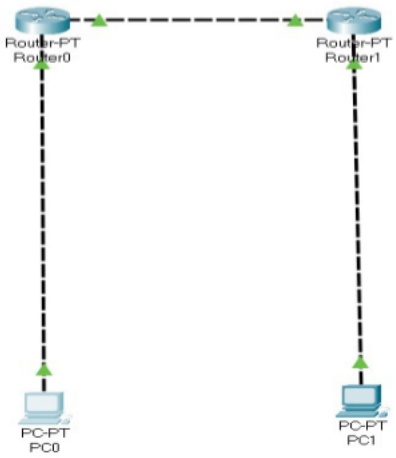
Conclusion :- The program was executed successfully

Practical No. 2
Study of Network Layer

a) Static Routing

Aim :- To study Static routing.

Circuit Diagram :-



Program :-
Router 0 > cli :- enable configure terminal interface fastethernet 0/0 ip address 10.0.0.1 255.0.0.0 no shutdown

exit interface fastethernet 0/1 ip address 20.0.0.1 255.0.0.0 no shutdown

exit ip route 40.0.0.0 255.0.0.0 20.0.0.2

Router 1 > cli :- enable configure terminal interface fastethernet 0/0 ip address 40.0.0.1 255.0.0.0 no shutdown

exit

interface fastethernet 0/1 ip address 20.0.0.2 255.0.0.0 no shutdown

exit ip route 10.0.0.0 255.0.0.0 20.0.0.1

Pc 0 > Desktop > Ip configuration :-

ipv4 : 10.0.0.2

subnet mask : 255.0.0.0

Default gateway: 10.0.0.1

Pc 1 > Desktop > Ip configuration :-

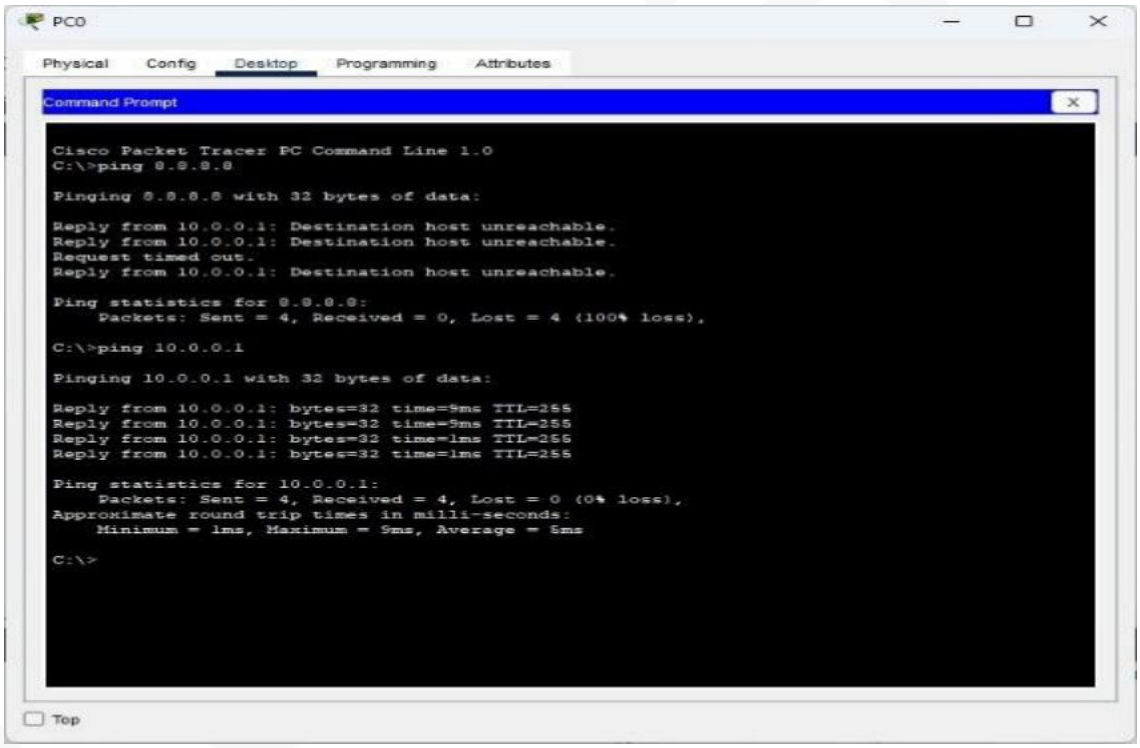
ipv4 : 40.0.0.2

subnet mask : 255.0.0.0

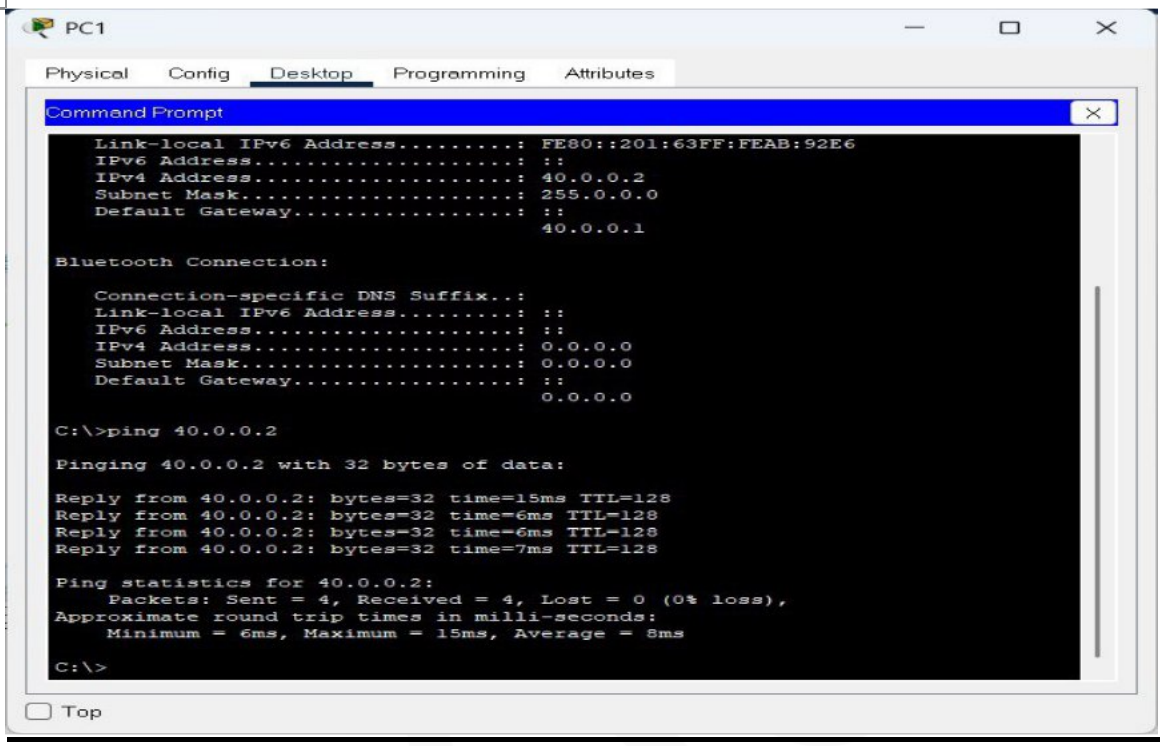
Default gateway: 40.0.0.1

Output :-

PC0 :-



PC1 :-

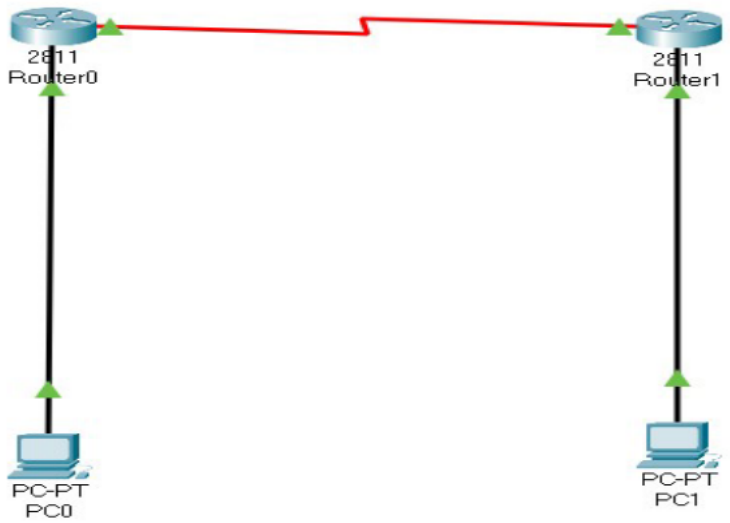


Conclusion :- The program was executed successfully.

b) RIP Routing

Aim :- To study RIP routing.

Circuit Diagram :-



Program :-
Router 0 > cli :- enable configure terminal interface fastethernet 0/0 ip address 10.0.0.1 255.0.0.0 no shutdown

exit interface fastethernet 0/1 ip address 20.0.0.1 255.0.0.0 no shutdown

exit router rip version 2
network 10.0.0.0 network 20.0.0.0 exit

Router 1 > cli :- enable configure terminal interface fastethernet 0/0 ip address 40.0.0.1 255.0.0.0 no shutdown

exit interface fastethernet 0/1 ip

address 20.0.0.2 255.0.0.0 no
shutdown

exit router rip version 2
network 40.0.0.0 network
20.0.0.0 exit

Pc 0 > Desktop > Ip configuration :- ipv4 :
10.0.0.2 subnet mask : 255.0.0.0

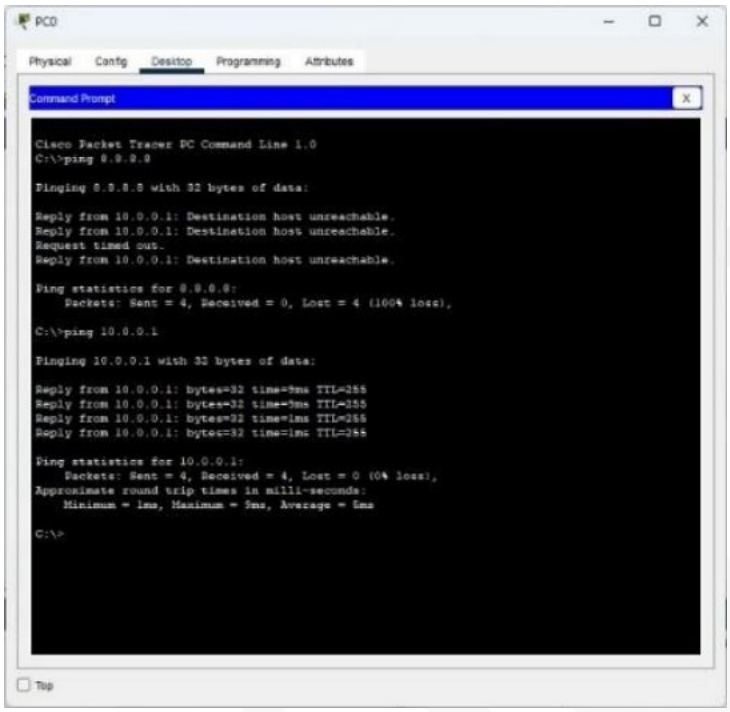
Default gateway: 10.0.0.1

Pc 1 > Desktop > Ip configuration :- ipv4 :
40.0.0.2

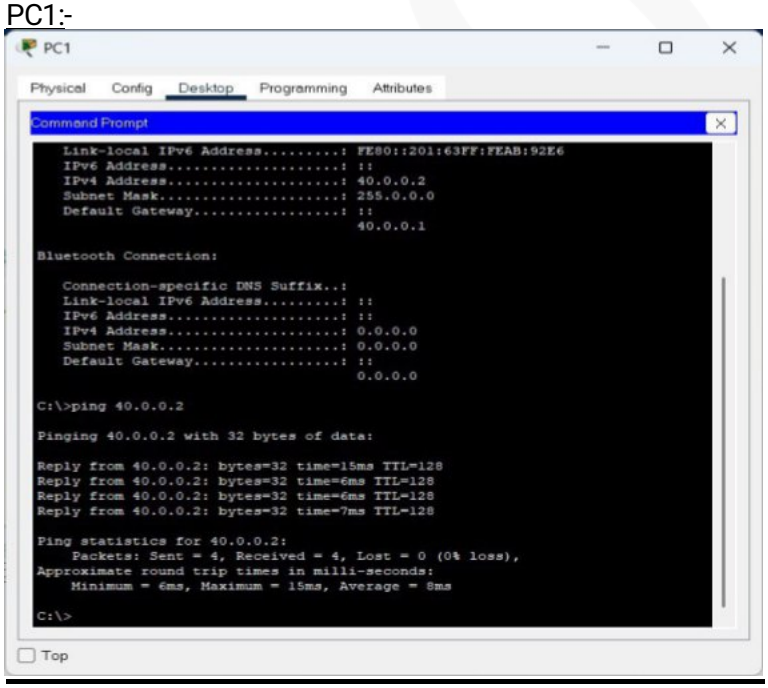
subnet mask : 255.0.0.0

Default gateway: 40.0.0.1

Output :-
PC0:-



PC1:-

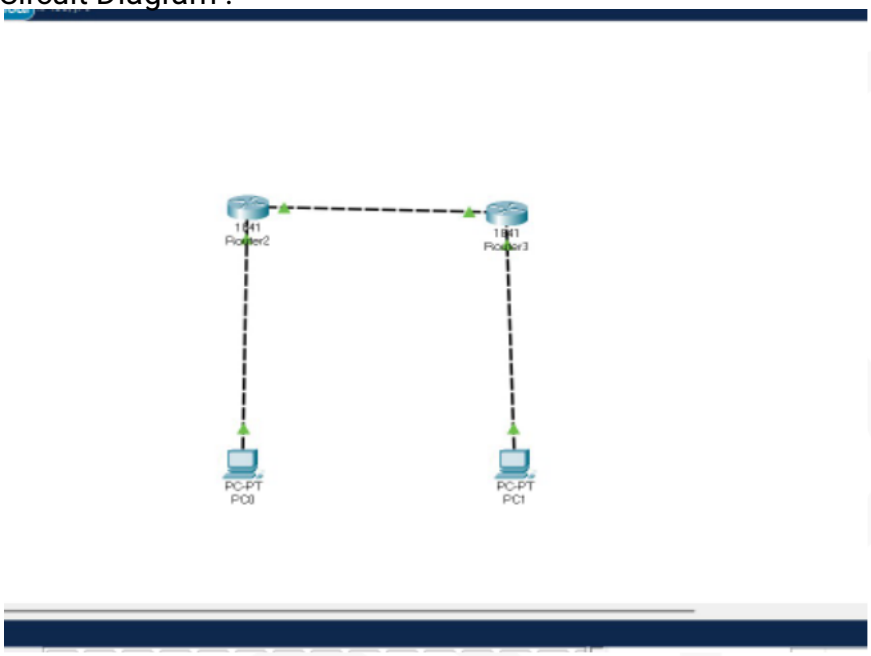


Conclusion :- The program was executed successfully.

c) OSPF Routing

Aim :- To study OSPF routing.

Circuit Diagram :-



Program :-
Router 0 > cli :- enable configure terminal interface fastethernet 0/0 ip address 10.0.0.1 255.0.0.0 no shutdown

exit interface fastethernet 0/1 ip address 20.0.0.1 255.0.0.0 no shutdown

exit router ospf 1 network 10.0.0.0 0.255.255.255 area 0 network 20.0.0.0 0.255.255.255 area 0 exit

Router 1 > cli :- enable configure terminal interface fastethernet 0/0 ip address 40.0.0.1 255.0.0.0 no shutdown

exit interface fastethernet 0/1 ip address 20.0.0.2 255.0.0.0 no shutdown

exit router ospf 2 network 40.0.0.0 0.255.255.255 area 0 network 20.0.0.0

0.255.255.255 area 0 exit

Pc 0 > Desktop > Ip configuration :- ipv4 :
10.0.0.2

subnet mask : 255.0.0.0

Default gateway: 10.0.0.1

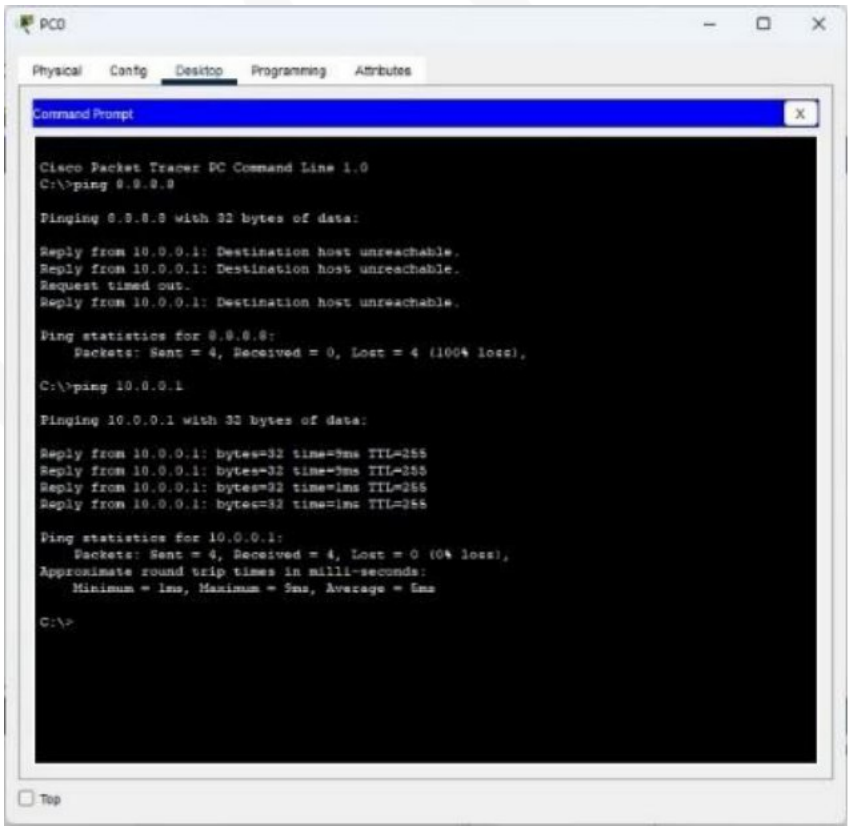
Pc 1 > Desktop > Ip configuration :- ipv4 :
40.0.0.2

subnet mask : 255.0.0.0

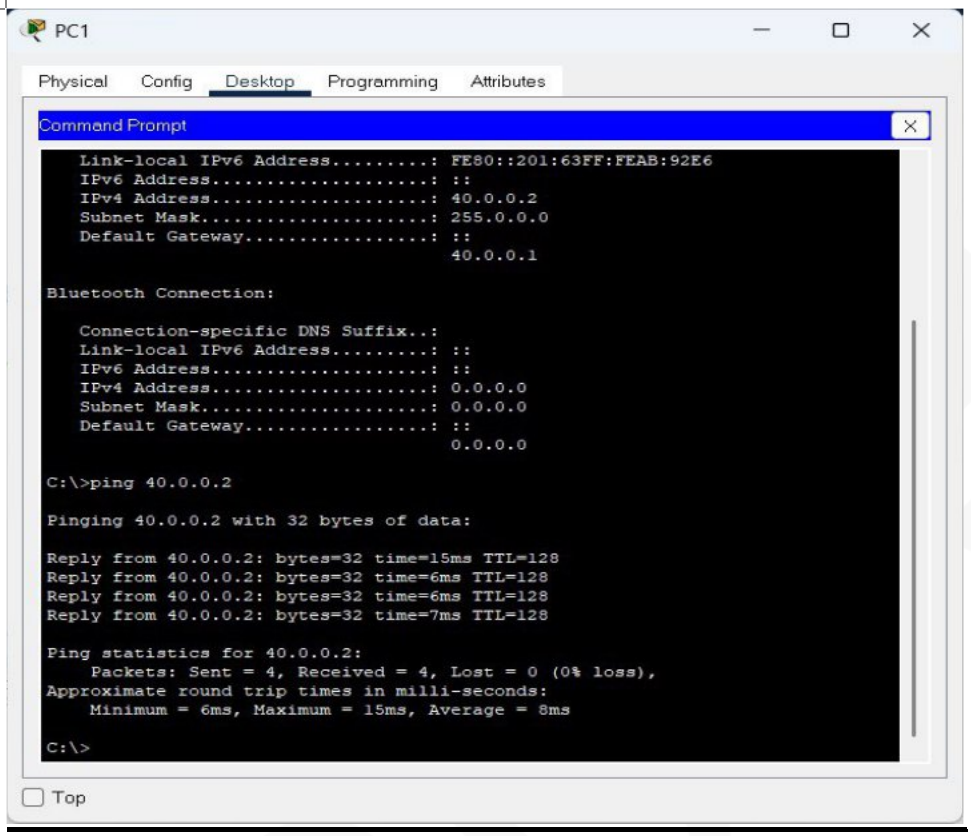
Default gateway: 40.0.0.1

Output :-

PC0:-



PC1:-



Conclusion :- The program was executed successfully.

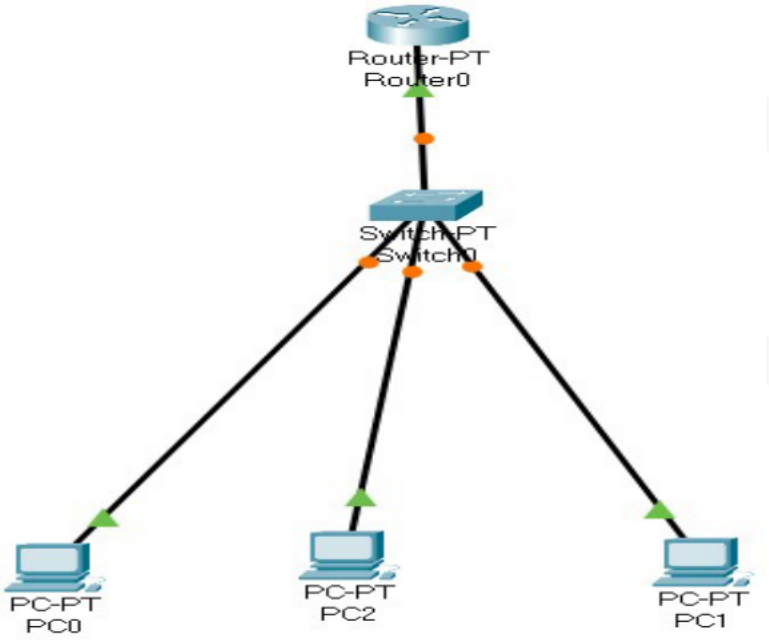
Practical No. 3

Study of Application Layer

a) DHCP

Aim :- To study DHCP.

Circuit Diagram : -



Program :-

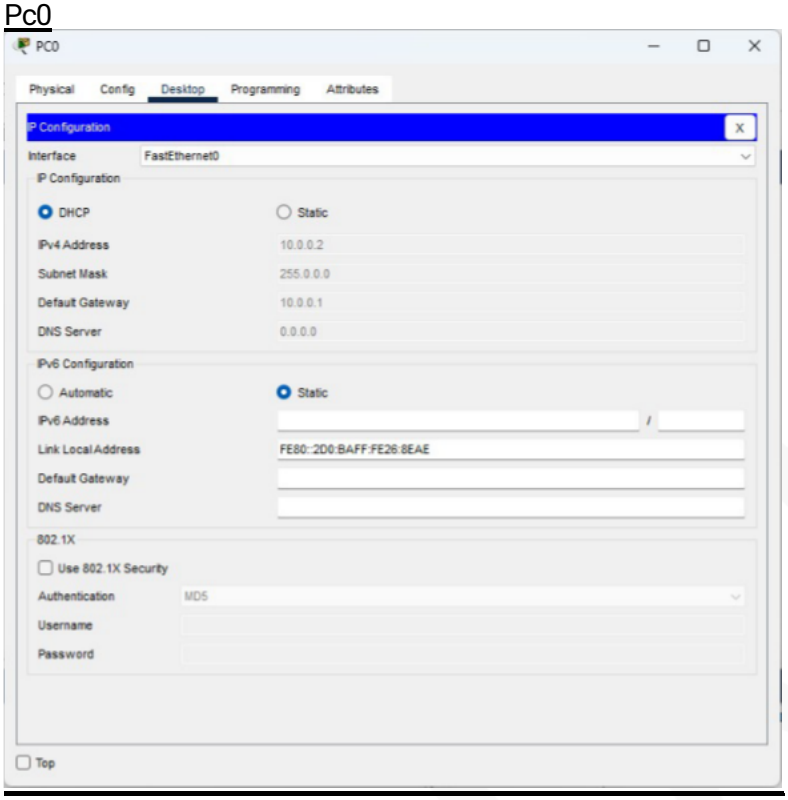
Router 0 > cli :- enable configure terminal interface fastethernet 0/0 ip address 10.0.0.1 255.0.0.0 no shutdown

exit ip dhcp pool syit network 10.0.0.0 255.0.0.0 default-router 10.0.0.1 ip dhcp excluded-address 10.0.0.2 10.0.0.1 ip dhcp pool syit network 10.0.0.0 255.0.0.0 default-router 10.0.0.1 ip dhcp excluded-address 10.0.0.2 10.0.0.13

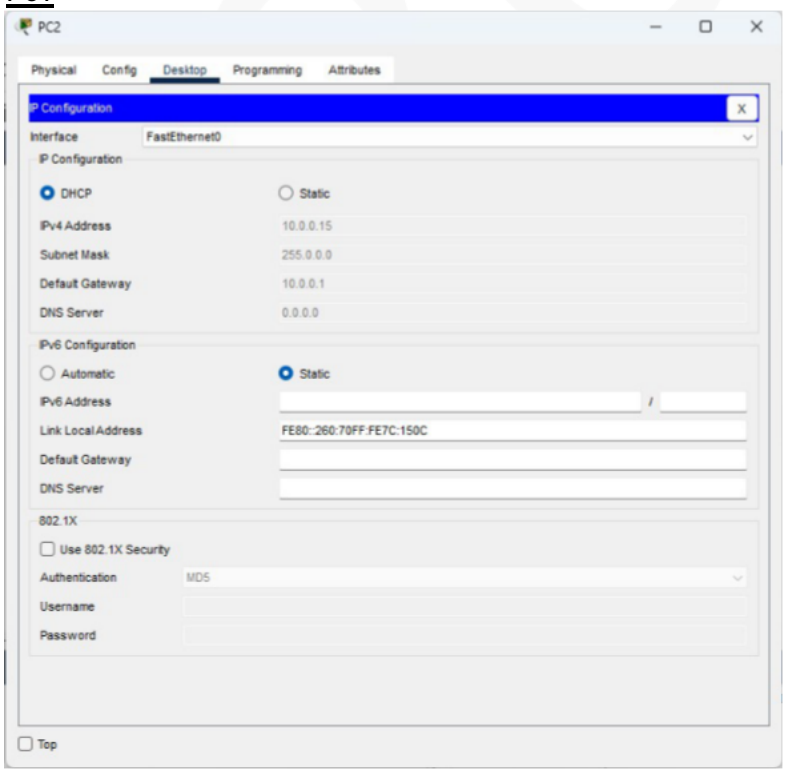
exit

Output:-

Pc0



Pc1

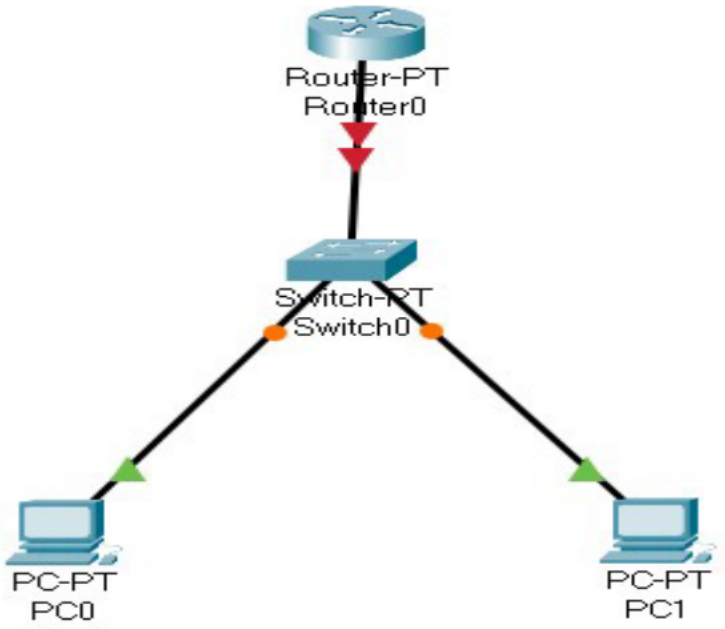


Conclusion :- The program was executed successfully.

b) DNS

Aim :- To study DNS.

Circuit Diagram :-



Program :-
Router 0 > cli :- enable configure terminal interface fastethernet 0/0 ip address 10.0.0.1 255.0.0.0 no shutdown

exit ip dhcp pool syit network 10.0.0.0 255.0.0.0 default-router 10.0.0.1 dns-server 8.8.8.8 exit ip host www.syit.com 10.0.0.1 exit

Pc 0 > Desktop > Ip configuration :- ipv4 : 10.0.0.2

subnet mask : 255.0.0.0

Default gateway: 10.0.0.1

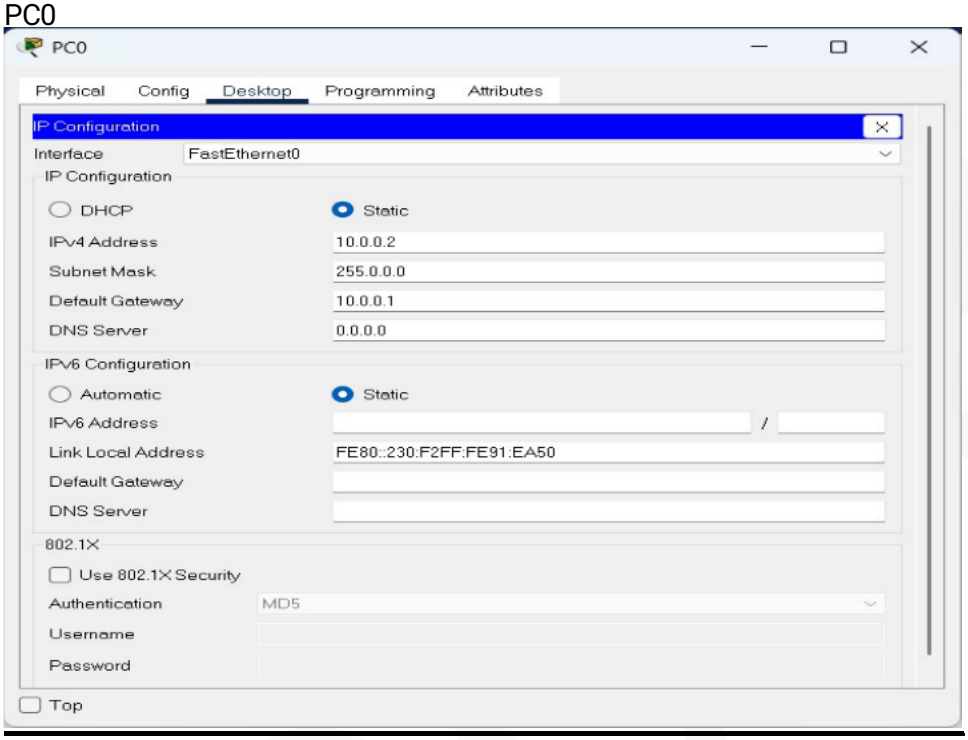
Pc 1 > Desktop > Ip configuration :- ipv4 : 10.0.0.3

subnet mask : 255.0.0.0

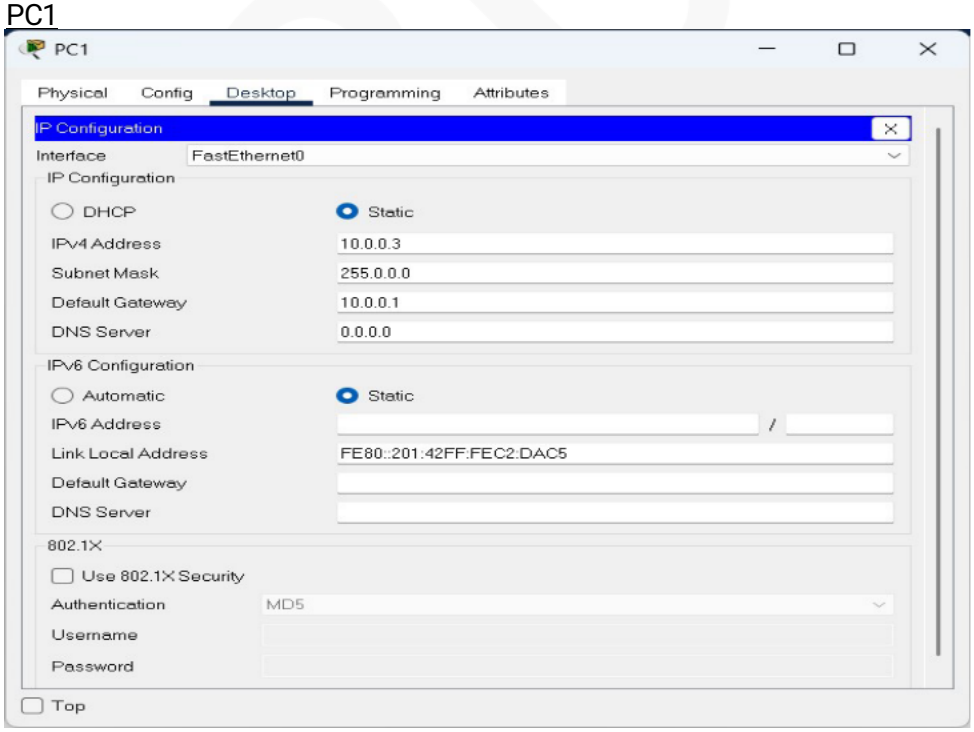
Default gateway: 10.0.0.1

Output:-

PC0



PC1

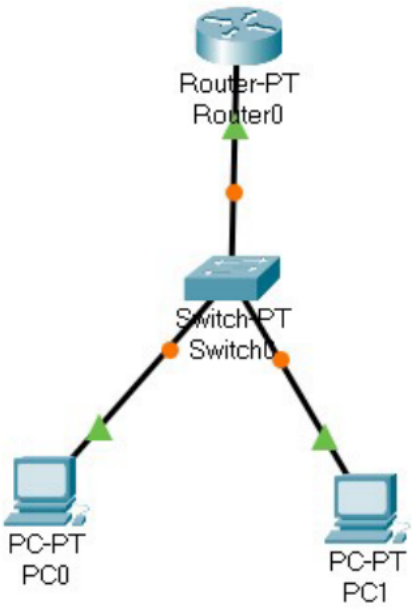


Conclusion:-The program executed Successfully

c) FTP

Aim :- To study FTP.

Diagram:



Program :-
Router 0 > cli :- enable configure terminal interface fastethernet 0/0 ip address 10.0.0.1 255.0.0.0 no shutdown

exit ip ftp username bscit ip ftp password syit ip ftp server enable exit

Pc 0 > Desktop > Ip configuration :- ipv4 : 10.0.0.2 subnet mask : 255.0.0.0

Default gateway: 10.0.0.1

Pc 1 > Desktop > Ip configuration :- ipv4 : 10.0.0.3

subnet mask : 255.0.0.0

Default gateway: 10.0.0.1

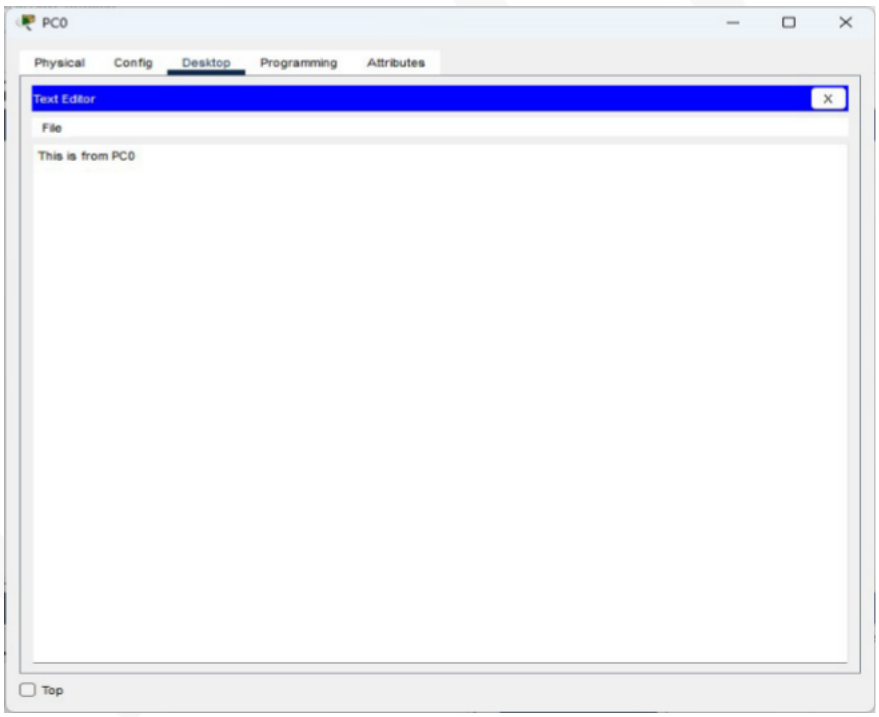
Pc 0 > Desktop > Text Editor :-

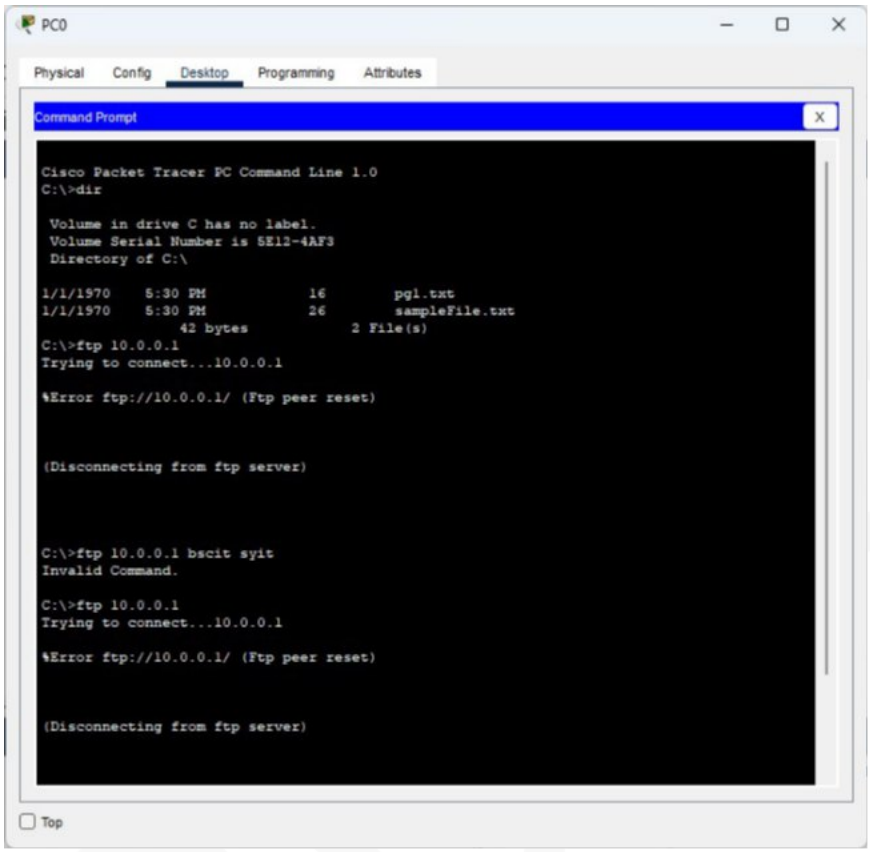
This is from PC0

Pc 0 > Desktop > Command Prompt :-
ftp 10.0.0.1 username : bscit
password : syit put
testfile.txt

Pc 1 > Desktop > Command Prompt :-
ftp 10.0.0.1

username : bscit
password : syit get
testfile.txt Output:-





Conclusion:-The program executed Successfully

d) HTTP

Aim :- To study HTTP.

Circuit Diagram:-

Program :- Server 0 > Desktop > Ip configuration :- ipv4 : 10.0.0.2 subnet mask : 255.0.0.0
Default gateway: 0.0.0.0

Server 0 > Services > HTTP :-
HTTP : on

HTTPS : on

Server 0 > Services > TFTP :- service: off

Server 0 > Services > HTTP > index.html > Edit :-
<html>

<head>

<title>SYIT</title>

</head>

<body>

<h1>I am SYIT</h1>

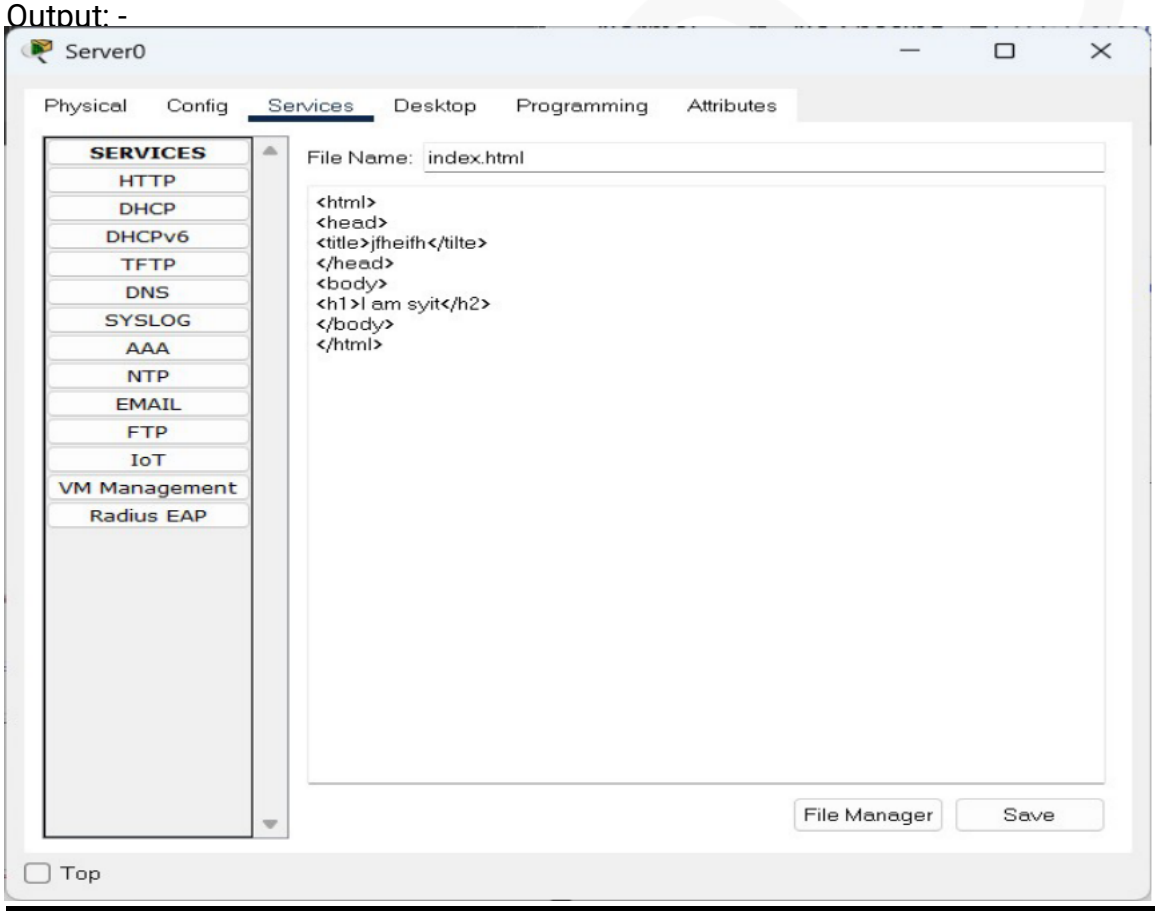
</body>

</html>

Pc 0 > Desktop > Ip Configuration :- ipv4 :
10.0.0.3 subnet mask : 255.0.0.0

Default gateway: 10.0.0.2

Output: -

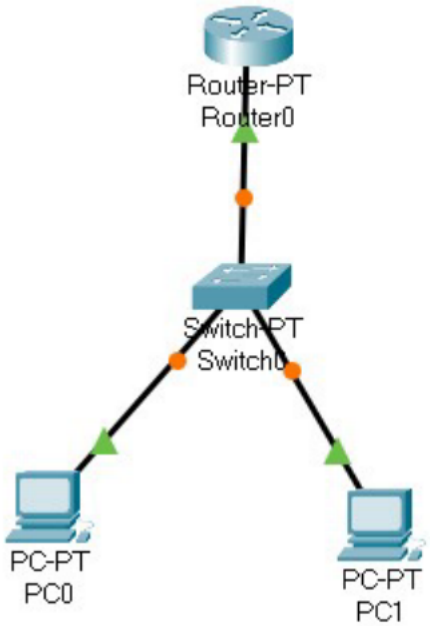


Conclusion:-The program executed Successfully.

e) TELNET

Aim :- To study TELNET.

Circuit diagram:-



Program :-
Router 0 > cli :- enable configure terminal interface fastethernet 0/0 ip address 10.0.0.1 255.0.0.0 no shutdown

exit line vty 0 10 password syit login transport input telnet exit exit

Pc 0 > Desktop > Ip configuration :- ipv4 : 10.0.0.2 subnet mask : 255.0.0.0

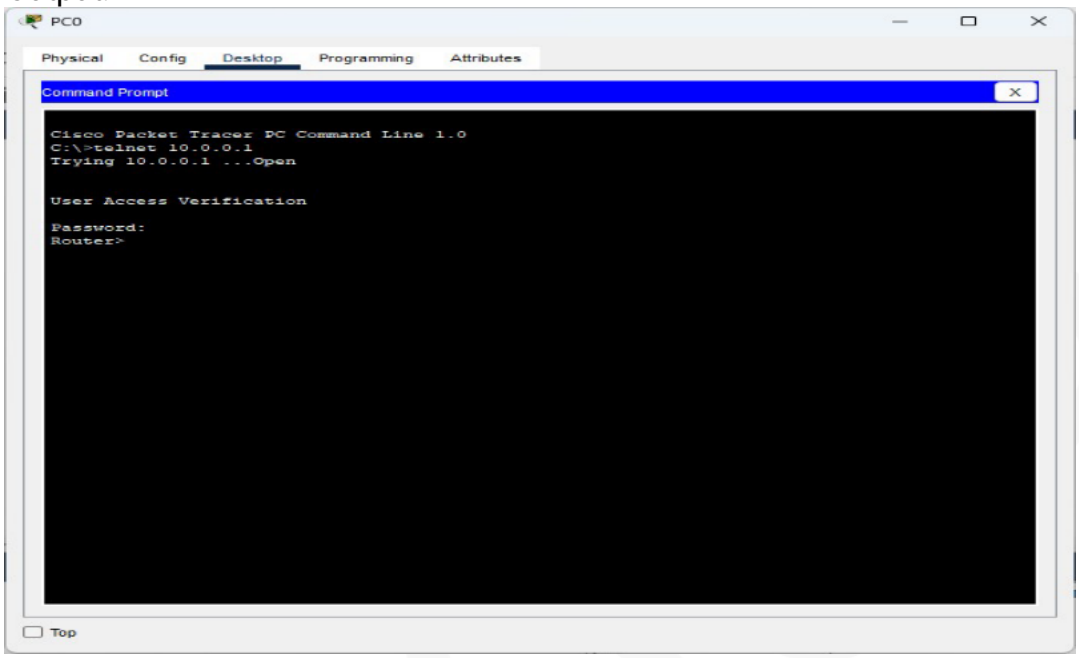
Default gateway: 10.0.0.1

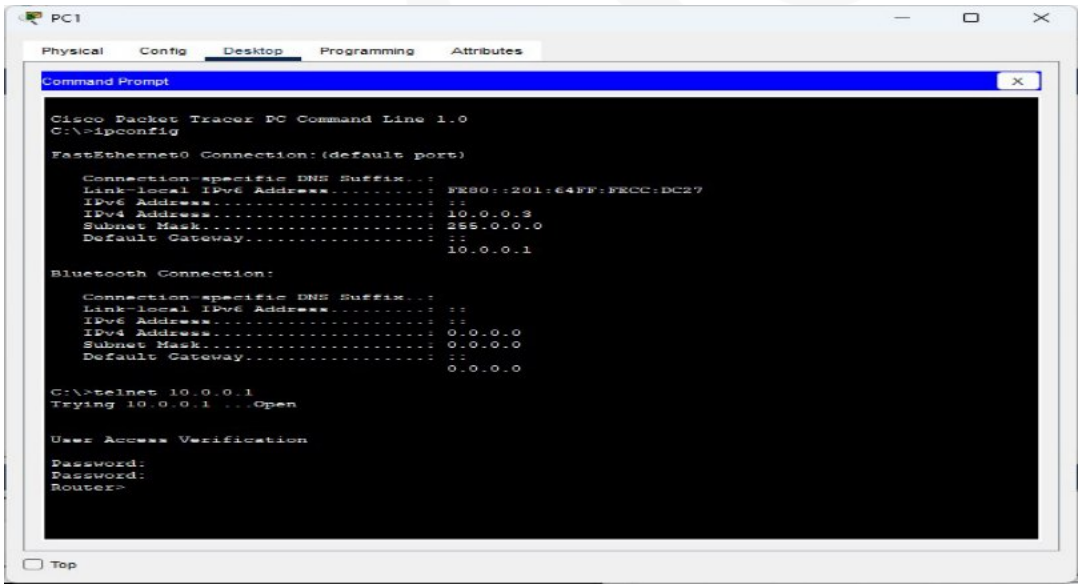
Pc 1 > Desktop > Ip configuration :- ipv4 : 10.0.0.3

subnet mask : 255.0.0.0

Default gateway: 10.0.0.1

Output:-





Conclusion:-The program executed Successfully.

Aim: To study and Implement Substitution Techniques in a Cryptography using python to perform Encryption & Description of a message.

Program:

```
def caesar_encrypt(text, shift):
 result = ""
  for char in text:
   if char.isalpha():
   base = ord('A') if char.isupper() else ord('a')
   result += chr((ord(char) - base + shift) % 26 + base)
    else:
    result += char
    return result


def caesar_decrypt(cipher, shift):

    return caesar_encrypt(cipher, -shift)

plaintext = "HELLO WORLD"

shift = 3

cipher = caesar_encrypt(plaintext, shift)

decrypted = caesar_decrypt(cipher, shift)


print("Ciphertext:", cipher)

print("Decrypted:", decrypted)
```

Output:

```
•••  Ciphertext: KHOOR ZRUOG
     Decrypted: HELLO WORLD
```

Conclusion: Substitution techniques in Python help demonstrate the basic principles of encryption by replacing characters with shifted or mapped values. While simple and easy to implement, these methods are not secure for real-world use but serve as a strong foundation for understanding modern cryptography.

Aim: To study and Implement Substitution Techniques in a Cryptography using python to perform Encryption & Description of a message.

Program:

```
from Cryptodome.Cipher import DES
from Cryptodome.Util.Padding import pad, unpad
from Cryptodome.Random import get_random_bytes
import binascii
data = b"DES is a symmetric key algorithm."
key = get_random_bytes(8)      # DES key must be exactly 8 bytes
iv = get_random_bytes(8)       # Initialization Vector (8 bytes)
cipher = DES.new(key, DES.MODE_CBC, iv)
ciphertext = cipher.encrypt(pad(data, DES.block_size)) # pad data to block size (8 bytes)
decipher = DES.new(key, DES.MODE_CBC, iv)
decrypted_data = unpad(decipher.decrypt(ciphertext), DES.block_size)
print("Original Data: ", data.decode())
print("Key (hex):     ", binascii.hexlify(key).decode())
print("IV (hex):      ", binascii.hexlify(iv).decode())
print("Ciphertext:    ", binascii.hexlify(ciphertext).decode())
print("Decrypted:     ", decrypted_data.decode())
```

Output:

```
Original Data:  DES is a symmetric key algorithm.
Key (hex):      8fb7cc5bb7d4f8db
IV (hex):       a5ca4e8101a6d0fe
Ciphertext:     cedf8fa2fbd6670def7e9554e307414f8206c25dd08d6ea0cdb1773d13275efde4e76ce6fee29a5f
Decrypted:      DES is a symmetric key algorithm.
```

Conclusion: DES shows how symmetric encryption works using a shared key. Though outdated today, it remains important for understanding the basics of block cipher design and modern encryption methods.

Aim: To Study and Implement Symmetric Key Cryptography using DES & AES Algorithm in Python.

Program:

```
from Cryptodome.Cipher import AES
from Cryptodome.Randomimport get_random_bytes
from Cryptodome.Util.Padding import pad, unpad
import base64

key = get_random_bytes(16) # 16 bytes (128-bit). You can also use 24 or 32.

def encrypt(plain_text: str) -> str:
    iv = get_random_bytes(AES.block_size)          # 16 bytes
    cipher = AES.new(key, AES.MODE_CBC, iv)
    ct_bytes = cipher.encrypt(pad(plain_text.encode('utf-8'), AES.block_size))
    # Prepend IV so decrypt or can extract it; encode as base64 for safe transport
    return base64.b64encode(iv + ct_bytes).decode('utf-8')

def decrypt(b64_ciphertext:str) -> str:
    data = base64.b64decode(b64_ciphertext)
    iv = data[:AES.block_size]
    ct = data[AES.block_size:]
    cipher = AES.new(key, AES.MODE_CBC, iv)
    pt = unpad(cipher.decrypt(ct), AES.block_size)
    return pt.decode('utf-8')

# Example usage
plain = "Hello, AES!"
ciphertext = encrypt(plain)
decrypted = decrypt(ciphertext)

print("Plain :", plain)
print("Cipher (b64):", ciphertext)
print("Decrypted:", decrypted)
```

output:

```
Plain : Hello, AES!
Cipher (b64): vrrRyPEPoJHHfkM0QNi/aCfLVn/uvSQkvWM+m/OyuhY=
Decrypted: Hello, AES!
```

Conclusion:

DES and AES demonstrate how symmetric encryption protects data using a shared key. While DES is outdated, AES remains a secure and efficient standard for modern cryptographic applications.

AIM :- To Study and Implement Asymmetric Key Cryptography using DH & RSA Algorithms in python.

Program:

```
# dh_example.py
from cryptography.hazmat.primitives.asymmetric import dh
from cryptography.hazmat.primitives.serialization import (
    Encoding, PublicFormat, load_pem_public_key, load_pem_parameters
)
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.kdf.hkdf import HKDF
from cryptography.hazmat.primitives.ciphers.aead import AESGCM
from cryptography.hazmat.primitives import serialization
import os

# 1) Create DH parameters (can be reused)
params = dh.generate_parameters(generator=2, key_size=2048)

a_private_key = params.generate_private_key()
a_public_key = a_private_key.public_key()
b_private_key = params.generate_private_key()
b_public_key = b_private_key.public_key()
a_pub_bytes = a_public_key.public_bytes(Encoding.PEM, PublicFormat.SubjectPublicKeyInfo)
b_pub_bytes = b_public_key.public_bytes(Encoding.PEM, PublicFormat.SubjectPublicKeyInfo)
```

```
a_pub_loaded = load_pem_public_key(a_pub_bytes)
b_pub_loaded = load_pem_public_key(b_pub_bytes)
```

```
# compute shared secrets
a_shared = a_private_key.exchange(b_pub_loaded) # bytes
b_shared = b_private_key.exchange(a_pub_loaded) # bytes

assert a_shared == b_shared # they must match

# Derive a symmetric key from the shared secret with HKDF
derived_key = HKDF(
    algorithm=hashes.SHA256(),
    length=32,          # 256-bit AES key
```

```
    salt=None,
    info=b"handshake data",
).derive(a_shared)

# Use AES-GCM for authenticated encryption
aesgcm = AESGCM(derived_key)
nonce = os.urandom(12)
plaintext = b"Secret message from A to B"
aad = b"optional associated data"

ciphertext = aesgcm.encrypt(nonce, plaintext, aad)
aesgcm_b = AESGCM(derived_key)
decrypted = aesgcm_b.decrypt(nonce, ciphertext, aad)
print("Plaintext:", plaintext)
print("Decrypted:", decrypted)
```

output:

```
Plaintext: b'Secret message from A to B'
Decrypted: b'Secret message from A to B'
```

Conclusion: DH and RSA demonstrate how asymmetric keys enable secure key exchange and

encryption, providing strong security for communication in modern systems.

AIM :- To study and implement the MDS (Message Digest 5) hashing algorithm using python.

Program:

```
import hashlib

text = "Hello MD5!"
hash_object = hashlib.md5(text.encode())
md5_hash = hash_object.hexdigest()

print("Original Text:", text)
print("MD5 Hash:", md5_hash)
```

output:

```
Original Text: Hello MD5!
MD5 Hash: 281d077ae77948c2c8533c01eeffd2d5
```

Conclusion:

MD5 hashing is simple to implement and useful for understanding basic hashing concepts, though it is not secure for modern applications.