Assignment No-5

Name- Prathamesh Gokulkar Rollno-33

Code

import sys

class Graph:

def init (self, nodes):

self.nodes = nodes # List of nodes

self.graph = {} # Dictionary to store adjacency list for graph

def add\_edge(self, u, v, weight): if u not in self.graph:

self.graph[u] = {} if v not in self.graph:

self.graph[v] = {} self.graph[u][v] = weight self.graph[v][u] = weight

def dijkstra(self, start\_node):

# Dictionary to store shortest path from start\_node to other nodes shortest\_paths = {node: sys.maxsize for node in self.nodes}

shortest\_paths[start\_node] = 0

# Dictionary to store the best previous node to get the shortest path previous\_nodes = {node: None for node in self.nodes}

visited = set()

nodes = self.nodes.copy()

while nodes:

# Get node with the smallest distance from start\_node min\_node = None

for node in nodes:

if node in visited:

continue

if min\_node is None:

min\_node = node

elif shortest\_paths[node] < shortest\_paths[min\_node]: min\_node = node

if min\_node is None:

break

# Nodes adjacent to the current node

neighbors = self.graph[min\_node].items()

for neighbor, weight in neighbors:

tentative\_value = shortest\_paths[min\_node] + weight if tentative\_value < shortest\_paths[neighbor]:

shortest\_paths[neighbor] = tentative\_value previous\_nodes[neighbor] = min\_node

visited.add(min\_node) nodes.remove(min\_node)

return previous\_nodes, shortest\_paths

# Function to display the shortest path from start\_node to each node def print\_shortest\_path(self, start\_node):

previous\_nodes, shortest\_paths = self.dijkstra(start\_node)

print(f"Shortest paths from {start\_node}:") for node in self.nodes:

if node == start\_node:

continue path = []

current\_node = node

while current\_node is not None:

path.append(current\_node)

current\_node = previous\_nodes[current\_node] path.reverse()

print(f"Path to {node}: {' -> '.join(path)}, Total cost: {shortest\_paths[node]}")

nodes = ['A', 'B', 'C', 'D', 'E']

graph = Graph(nodes) graph.add\_edge('A', 'B', 4)

graph.add\_edge('A', 'C', 2)

graph.add\_edge('B', 'C', 1)

graph.add\_edge('B', 'D', 5)

graph.add\_edge('C', 'D', 8)

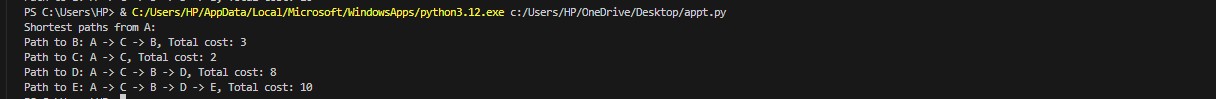
graph.add\_edge('C', 'E', 10)

graph.add\_edge('D', 'E', 2)

start\_node = 'A'

graph.print\_shortest\_path(start\_node)

output:



Assignment No-3

Name- Prathamesh Gokulkar Rollno-33

Code

import ipaddress

def calculate\_subnet(network, subnet\_bits):

try:

# Convert the network to an IPv4Network object net = ipaddress.IPv4Network(network)

# Calculate the new subnet mask

new\_prefix = net.prefixlen + subnet\_bits if new\_prefix > 32:

raise ValueError("Invalid subnet bits. Subnet prefix length cannot exceed 32.")

# Create subnets

subnets = list(net.subnets(new\_prefix=new\_prefix))

# Display results

print(f"Original Network: {network}")

print(f"Original Subnet Mask: {net.netmask}")

print(f"New Subnet Mask: {subnets[0].netmask}") print(f"Number of Subnets: {len(subnets)}")

print(f"Hosts per Subnet: {subnets[0].num\_addresses - 2}") # Exclude network and broadcast addresses

print("\nSubnets:") for subnet in subnets:

print(subnet)

except ValueError as e: print(f"Error: {e}")

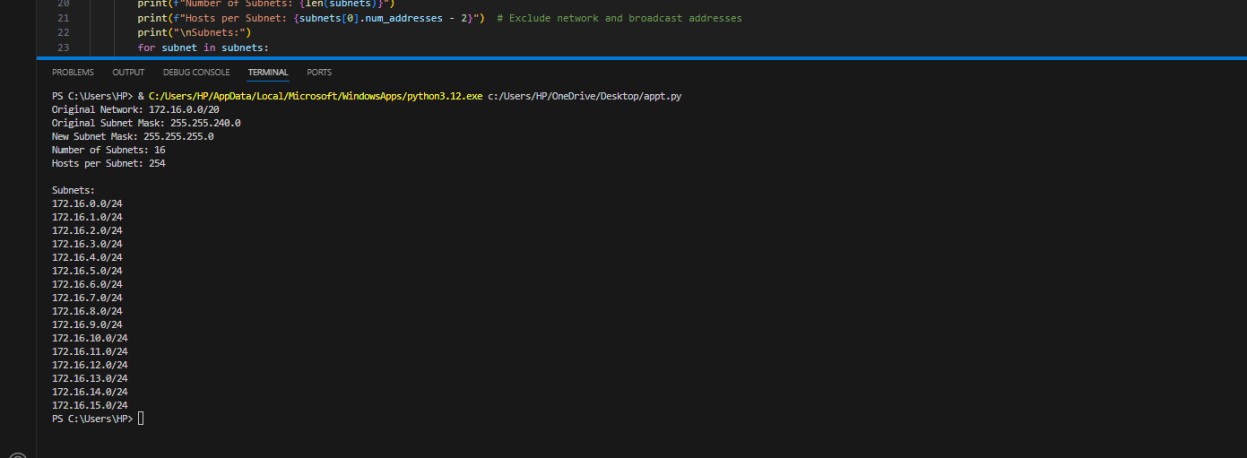
if name == " main ":

# Example usage

network = "172.16.0.0/20" # Define the network in CIDR notation subnet\_bits = 4 # Number of bits to use for subnetting

calculate\_subnet(network, subnet\_bits)

ouput:



Assignment No-9

Name- Prathamesh Gokulkar Rollno-33

Code

import socket

def dns\_lookup(ip\_address=None, domain\_name=None):

try:

if ip\_address:

# Perform reverse DNS lookup

result = socket.gethostbyaddr(ip\_address)

return f"IP Address: {ip\_address} -> Domain Name: {result[0]}" elif domain\_name:

# Perform forward DNS lookup

result = socket.gethostbyname(domain\_name)

return f"Domain Name: {domain\_name} -> IP Address: {result}" else:

return "Please provide either an IP address or a domain name." except socket.herror as e:

return f"Error: Unable to resolve {ip\_address or domain\_name}. {e}" except socket.gaierror as e:

return f"Error: {e}"

if name == " main ":

print("DNS Lookup Program")

# Input from user

choice = input("Do you want to lookup by (1) IP Address or (2) Domain Name? Enter 1 or 2: ")

if choice == "1":

ip\_address = input("Enter the IP address: ") result = dns\_lookup(ip\_address=ip\_address) print(result)

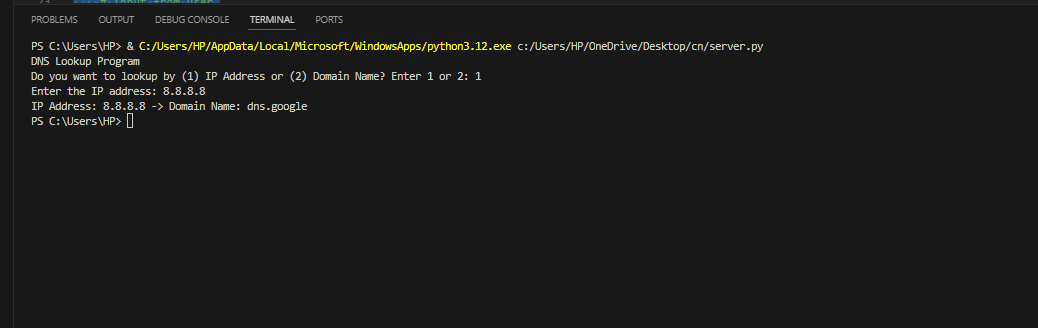
elif choice == "2":

domain\_name = input("Enter the domain name: ") result = dns\_lookup(domain\_name=domain\_name) print(result)

else:

print("Invalid choice. Please run the program again.")

OUPUT:



Assignment No-7

Name- Prathamesh Gokulkar Rollno-33

Code

Server.py

import socket

def send\_file(filename, conn):

with open(filename, "rb") as file: while chunk := file.read(1024):

conn.send(chunk)

print(f"File {filename} sent successfully!")

def start\_server(host='127.0.0.1', port=65432):

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) server\_socket.bind((host, port))

server\_socket.listen(1)

print(f"Server listening on {host}:{port}...")

conn, addr = server\_socket.accept() print(f"Connected to {addr}")

filename = conn.recv(1024).decode()

try:

send\_file(filename, conn) except FileNotFoundError:

print(f"File {filename} not found.") conn.send(b"ERROR: File not found.")

conn.close()

if name == " main ":

start\_server()

client.py

import socket

def receive\_file(filename, host='127.0.0.1', port=65432):

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) client\_socket.connect((host, port))

client\_socket.send(filename.encode())

with open("received\_" + filename, "wb") as file: while True:

data = client\_socket.recv(1024) if not data:

break file.write(data)

print(f"File {filename} received and saved as received\_{filename}") client\_socket.close()

if name == " main ":

filename = input("Enter the filename to download: ") receive\_file(filename)

ouput:

C:\Users\HP\OneDrive\Desktop\cn>python server.py Server listening on 127.0.0.1:65432...

Connected to ('127.0.0.1', 52151) File tr.txt sent successfully!

C:\Users\HP\OneDrive\Desktop\cn>python client.py Enter the filename to download: tr.txt

File tr.txt received and saved as received\_tr.txt

Assignment No-6

Name- Prathamesh Gokulkar Rollno-33

Code Server

import socket

def start\_server():

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) server\_socket.bind(('localhost', 12345))

server\_socket.listen(1)

print("Server is listening for connections...")

while True: try:

client\_socket, addr = server\_socket.accept()

print(f"Connection from {addr} has been established!")

data = client\_socket.recv(1024) if data:

print(f"Received from client: {data.decode('utf-8')}") client\_socket.sendall(b"Hello, Client!")

else:

print("No data received from client.")

client\_socket.close() except Exception as e:

print(f"Error: {e}")

if name == " main ":

start\_server()

client

import socket

def start\_client():

try:

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) print("Connecting to server...")

client\_socket.connect(('localhost', 12345)) print("Connected to server.")

client\_socket.sendall(b"Hello, Server!") print("Message sent to server.")

data = client\_socket.recv(1024)

print(f"Received from server: {data.decode('utf-8')}")

except Exception as e: print(f"Error: {e}")

finally:

client\_socket.close()

print("Client socket closed.")

if name == " main ": start\_client()

ouput:

C:\Users\HP\OneDrive\Desktop\cn>python server.py Server is listening for connections...

Connection from ('127.0.0.1', 52095) has been established! Received from client: Hello, Server!

C:\Users\HP\OneDrive\Desktop\cn>python client.py Connecting to server...

Connected to server. Message sent to server.

Received from server: Hello, Client! Client socket closed.

Assignment no-8

Name- Prathamesh Gokulkar Rollno-33

