ARTIFICIAL INTELLIGENCE

Task - 1

PART- B

R2:

Source Code:

```
import networkx as nx
import random

G = nx.erdos_renyi_graph(100, 0.03)

fleet = [[] for _ in range(30)]

clock_ticks = 600

customer_counter = 0

print("Customer  Pick-upNode  DropNode  AvgDistance  AvgTime")
for tick in range(clock_ticks):

    if random.random() < 0.3:
        customer_counter += 1
        customer = f"C{customer_counter}"


        valid_nodes = list(G.nodes())
        pickup_node = random.choice(valid_nodes)
        valid_nodes.remove(pickup_node)
        dropoff_node = random.choice(valid_nodes)


        if nx.has_path(G, pickup_node, dropoff_node):
            path = nx.shortest_path(G, source=pickup_node, target=dropoff_node)
            avg_distance = len(path) - 1
            avg_time = avg_distance
            print(f"{customer}          {pickup_node}          {dropoff_node}          {avg_distance}
     {avg_time}")
        else:
            print(f"{customer}          {pickup_node}          {dropoff_node}          No path found")
```

Output:

Customer	Pick-upNode	DropNode	AvgDistance	AvgTime
C1	89	7	4	4

C2	77	62	5	5
C3	9	44	4	4
C4	54	27	4	4
C5	73	70	4	4
C6	88	97	4	4
C7	99	5	3	3
C8	64	88	3	3
C9	64	47	3	3
C10	76	30	4	4
C11	16	67	2	2
C12	62	68	3	3
C13	35	8	5	5
C14	80	76	4	4
C15	1	44	6	6
C16	25	33	3	3
C17	64	89	5	5
C18	44	71	5	5
C19	30	81	4	4
C20	54	75	5	5
C21	13	32	4	4
C22	16	87	4	4
C23	42	94	7	7
C24	27	45	4	4
C25	28	6	4	4
C26	85	2	4	4
C27	58	1	No path found	
C28	2	33	4	4
C29	31	18	4	4
C30	29	57	3	3
C31	90	50	2	2
C32	40	42	4	4
C33	72	56	2	2
C34	13	64	4	4
C35	58	77	No path found	
C36	61	64	2	2
C37	31	56	2	2
C38	48	89	5	5
C39	36	69	6	6
C40	6	30	1	1
C41	54	28	3	3

C42	3	83	4	4
C43	52	71	5	5
C44	41	39	2	2
C45	56	48	4	4
C46	84	73	3	3
C47	82	67	3	3
C48	30	13	4	4
C49	81	70	2	2
C50	16	55	7	7
C51	37	71	3	3
C52	79	12	4	4
C53	4	14	4	4
C54	2	1	6	6
C55	13	8	2	2
C56	70	83	2	2
C57	36	2	6	6
C58	99	24	5	5
C59	82	45	3	3
C60	20	75	5	5
C61	73	58	No path found	
C62	6	58	No path found	
C63	66	21	No path found	
C64	17	46	No path found	
C65	65	70	5	5
C66	48	15	5	5
C67	0	86	5	5
C68	12	26	4	4
C69	78	45	5	5
C70	8	19	2	2
C71	98	93	5	5
C72	23	75	5	5
C73	97	9	4	4
C74	47	83	4	4
C75	17	3	5	5
C76	13	5	4	4
C77	29	25	4	4
C78	47	67	2	2
C79	37	32	3	3
C80	61	7	3	3
C81	47	19	3	3

C82	6	26	5	5
C83	77	23	5	5
C84	59	30	3	3
C85	51	2	5	5
C86	19	13	4	4
C87	22	64	2	2
C88	2	32	5	5
C89	78	22	5	5
C90	59	91	4	4
C91	32	37	3	3
C92	28	32	2	2
C93	13	98	5	5
C94	38	10	6	6
C95	51	78	4	4
C96	35	6	5	5
C97	71	5	4	4
C98	33	43	4	4
C99	85	72	2	2
C100	49	81	3	3
C101	17	73	4	4
C102	75	58	No path found	
C103	75	94	4	4
C104	14	92	1	1
C105	28	31	5	5
C106	22	72	3	3
C107	34	87	4	4
C108	2	94	3	3
C109	69	77	3	3
C110	88	41	3	3
C111	71	65	3	3
C112	17	13	3	3
C113	48	18	2	2
C114	69	26	3	3
C115	63	2	3	3
C116	84	8	3	3
C117	3	90	1	1
C118	74	80	4	4
C119	23	71	5	5
C120	27	61	5	5
C121	32	36	5	5

C122	44	39	5	5
C123	8	58	No path found	
C124	49	50	3	3
C125	69	25	5	5
C126	67	59	2	2
C127	72	69	2	2
C128	73	4	5	5
C129	61	79	1	1
C130	93	16	2	2
C131	87	61	4	4
C132	78	65	3	3
C133	37	71	3	3
C134	36	27	5	5
C135	2	39	5	5
C136	66	31	6	6
C137	37	20	4	4
C138	25	74	4	4
C139	48	77	5	5
C140	26	63	5	5
C141	96	53	3	3
C142	64	29	5	5
C143	85	97	4	4
C144	53	70	5	5
C145	42	89	5	5
C146	59	61	4	4
C147	42	84	3	3
C148	17	59	3	3
C149	17	66	2	2
C150	17	83	4	4
C151	99	23	4	4
C152	43	84	3	3
C153	48	23	3	3
C154	91	89	4	4
C155	29	18	4	4
C156	85	52	4	4
C157	94	4	5	5
C158	40	13	3	3
C159	72	83	3	3
C160	44	60	6	6
C161	1	73	5	5

C162	96	4	5	5
C163	69	35	2	2
C164	0	52	3	3
C165	67	75	4	4
C166	14	32	3	3
C167	37	96	5	5
C168	86	20	5	5
C169	21	55	No path found	
C170	50	27	2	2
C171	90	6	3	3
C172	20	18	3	3
C173	67	69	3	3
C174	78	17	4	4
C175	30	39	5	5
C176	32	36	5	5
C177	5	59	3	3
C178	51	79	5	5
C179	92	88	4	4
C180	87	61	4	4
C181	41	99	4	4
C182	90	72	4	4
C183	66	98	5	5
C184	50	79	3	3
C185	89	67	4	4
C186	32	92	2	2
C187	97	86	6	6
C188	66	50	3	3
C189	28	33	1	1
C190	85	68	3	3
C191	96	17	6	6
C192	44	26	6	6
C193	73	46	No path found	
C194	75	38	6	6
C195	36	82	4	4
C196	27	45	4	4
C197	84	4	4	4
C198	61	72	4	4
C199	17	68	4	4
C200	22	27	3	3
C201	80	18	5	5

C202 34 98 5 5

R3:

Source Code:

```
import networkx as nx
import random

G = nx.random_regular_graph(3, 100)

fleet = [[] for _ in range(30)]

def find_shortest_path(vehicle_id):
    current_position = fleet[vehicle_id][-1] if fleet[vehicle_id] else
random.choice(list(G.nodes()))
    target = random.choice(list(G.nodes()))
    while target == current_position:
        target = random.choice(list(G.nodes()))
    path = nx.shortest_path(G, source=current_position, target=target)
    return path

total_distance = 0
total_trips = 0
hours_per_day = 24

for hour in range(hours_per_day):
    reservations = random.randint(450, 600)

    for _ in range(reservations):
        vehicle_id = random.randint(0, 29)
        path = find_shortest_path(vehicle_id)
        fleet[vehicle_id] += path
        total_distance += len(path) - 1
        total_trips += 1

average_distance_per_day = total_distance / hours_per_day
```

```
average_trips_per_day = total_trips / hours_per_day

print(f"Average Distance Traveled per Day: {average_distance_per_day:.2f} edges")
print(f"Average Number of Trips per Day: {average_trips_per_day:.2f} trips")
```

Output:
Average Distance Traveled per Day: 2570.29 edges
Average Number of Trips per Day: 531.79 trips

Answer:
This code attempted to build upon the Python software created in response to Requirement (R2) by implementing three significant changes. The fixed-size road network was to be replaced with an additional dynamic random graph with 100 nodes & an average connectivity of 3. The fleet size would also be increased, going from two to thirty cars. In order to simulate the unpredictable nature of real-world demand, they had to change the reservation rate to cover a range of 450 to 600 reservations per hour. The Python software was intentionally changed to accomplish these goals. Using the NetworkX package, which produces a network with 100 nodes and an average degree of connectedness set to 3, made it easier to create the random graph. The simulation was changed to include diversity, which increased the simulation's realism of the road network. The program expanded to manage a total of 30 cars as the fleet size increased. To ensure flawless coordination & assignment of pickups and drop-offs, this needed adjustments to data structures and operations. The adjustment that affected reservation generation may have been the most significant. The simulation was built to produce a random number of reservations every hour, precisely ranging between 450 & 600 reservations, using the updated reservation rate settings. Each reservation had a unique vehicle assigned to it, which set off a series of network-wide path calculations & vehicle movements. The results of running the updated software showed the average daily mileage of the entire fleet and the typical number of trips made each day. These metrics offered quantifiable information to assess the effectiveness of the adjustments made in R3 and provided helpful insights into the efficiency and effectiveness of the simulated fleet. Requirement (R3) significantly improved the simulation by adding unpredictability to the road network, increasing the number of the fleet, and adding variation in reservation rates (Farhan,

2019). These adjustments improved the simulation's realism and ability to adjust to changing demand, producing useful measures to evaluate fleet performance in a changing environment.

R4:

Source Code:

```
import networkx as nx
import random

G = nx.random_regular_graph(3, 100)

fleet = [[] for _ in range(60)]

def find_shortest_path(vehicle_id, current_position):
    target = random.choice(list(G.nodes()))
    while target == current_position:
        target = random.choice(list(G.nodes()))
    path = nx.shortest_path(G, source=current_position, target=target)
    return path

total_distance = 0
total_trips = 0
hours_per_day = 24

for hour in range(hours_per_day):
    reservations = random.randint(450, 600)

    for _ in range(reservations):
        vehicle_id = random.randint(0, 59)
        current_position = fleet[vehicle_id][-1] if fleet[vehicle_id] else
random.choice(list(G.nodes()))
        path = find_shortest_path(vehicle_id, current_position)
        fleet[vehicle_id] += path
```

```
        total_distance += len(path) - 1
        total_trips += 1

average_distance_per_day = total_distance / hours_per_day

average_trips_per_day = total_trips / hours_per_day

print(f"Average Distance Traveled per Day: {average_distance_per_day:.2f} edges")
print(f"Average Number of Trips per Day: {average_trips_per_day:.2f} trips")
```

Output:
Average Distance Traveled per Day: 2587.88 edges
Average Number of Trips per Day: 531.04 trips

Answer:
The code is significantly expanded the Python software created in response to Requirement (R3) in order to meet Requirement (R4). Notably, it maintained the same reservation rate characteristics while scaling up the fleet size significantly to accommodate a fleet of 60 vehicles. The project's success provided insightful information on the increased fleet's performance and operational effectiveness. The program changes were carried out methodically. In order to manage the complexity of a bigger vehicle ensemble, data systems, and operations were modified to facilitate the administration and coordination of a larger fleet. This scalability reflected real-world situations in which larger fleets frequently face strong demand and a variety of operational difficulties. The program maintained adherence to the reservation rate guidelines established in R3 despite the significant increase in fleet size. Every hour, reservations continued to be generated at random times while maintaining a fluctuating range of 450 to 600 reservations. To start the process of path computations and vehicle movements inside the network, each reservation was carefully assigned to a particular vehicle.

The results of running the modified program were both illuminating and educational. The huge 60-vehicle fleet's daily average distance traveled was calculated to be around 2587.88 edges. The fleet's regular operational effectiveness in negotiating the dynamic road network was greatly influenced by this measurement (Chasserieau and Goix, 2019). Furthermore, it was found that 531.04 trips were completed on average each day. This metric gave a concrete evaluation of how

efficiently the fleet served passengers throughout the simulated environment. Requirement (R4), which included a larger fleet while maintaining the identical reservation rate dynamics developed in R3, effectively reflected a significant advancement in the simulation. The obtained data demonstrated the fleet's ability to effectively handle a rise in demand, illuminating the scalability and operational effectiveness of the simulated transportation system.

R5:

Source Code:

```
import networkx as nx
import random

G = nx.random_regular_graph(4, 100)

fleet = [[] for _ in range(60)]

def find_shortest_path(vehicle_id, current_position):
    target = random.choice(list(G.nodes()))
    while target == current_position:
        target = random.choice(list(G.nodes()))
    path = nx.shortest_path(G, source=current_position, target=target)
    return path

total_distance = 0
total_trips = 0
hours_per_day = 24

for hour in range(hours_per_day):
    reservations = random.randint(450, 600)

    for _ in range(reservations):
        vehicle_id = random.randint(0, 59)
```

```
        current_position = fleet[vehicle_id][-1] if fleet[vehicle_id] else
random.choice(list(G.nodes()))
        path = find_shortest_path(vehicle_id, current_position)
        fleet[vehicle_id] += path
        total_distance += len(path) - 1
        total_trips += 1

average_distance_per_day = total_distance / hours_per_day

print(f"New Average Distance Traveled per Day with Connectivity 4:
{average_distance_per_day:.2f} edges")
```

Output:
New Average Distance Traveled per Day with Connectivity 4: 1873.38 edges

Answer:
The code has been changed to the Python program that had been previously modified to satisfy the requirements listed in Requirement (R4) in response to Requirement (R5). The main goal was to determine how better network connectivity would affect a simulated fleet's operational efficiency. The program modification process was meticulously carried out. By increasing the network's connectivity from 2 to 4, the graph became denser as well as intricately interwoven. With the introduction of numerous new alternate routes and roads, this modification dramatically altered the navigational scene. The fleet's operations were expected to become more efficient overall because of these new dynamics, which also had the possibility to shorten journey times (Zhang *et al.* 2022). The findings of running the modified program were both instructive and suggestive of the effects of increased connection. The new average distance traveled per day with connectivity 4 was calculated by the application. This calculation yielded a number of roughly 1873.38 edges, which may be used as a quantitative measure of the fleet's average daily travel distance under the effect of the more dense network structure.

When comparing the findings of this new statistic to those from Requirement (R4), it was apparent that the average distance was significantly lower. This decrease in typical distance served as evidence of the significant influence network architecture has on fleet performance

(Rapelli *et al.* 2019). Greater network connectivity allowed for more effective routing and reduced journey times. Vehicles could move across the network more quickly and easily because there were more direct routes and options for driving. By raising the network's connectivity to 4, the Requirement (R5) brought a fundamental change that allowed the Python software to calculate the new average distance traveled each day with connectivity 4. The fact that this metric's value was lower than R4's illustrated the real advantages of a denser road network, wherein reduced travel times and better routing led to increased operational effectiveness for the fictitious fleet.