# TASK 1:

class Bus:  
 def \_\_init\_\_(self, capacity=34, female\_capacity=17, male\_capacity=17):  
 self.capacity = capacity  
 self.capacity\_added=0  
 self.female\_capacity = female\_capacity  
 self.male\_capacity = male\_capacity  
 self.female\_added = 0  
 self.male\_added = 0  
 self.students = []  
  
 def add\_student(self, name, gender):  
 if self.capacity\_added<self.capacity:  
 if gender == 'female' and self.female\_added < self.capacity:  
 seat\_number = self.female\_added + 1  
 self.students.append({'name': name, 'gender': 'female', 'seat\_number': seat\_number})  
 print(f"{name} added to the bus. Seat number: {seat\_number}")  
 self.female\_added += 1  
 self.capacity\_added += 1  
 elif gender == 'male' and self.male\_added < self.male\_capacity:  
 seat\_number = self.male\_added + 1  
 self.students.append({'name': name, 'gender': 'male', 'seat\_number': seat\_number})  
 print(f"{name} added to the bus. Seat number: {seat\_number}")  
 self.male\_added += 1  
 self.capacity\_added += 1  
  
 else:  
 print(f"Cannot add {name}. No more seats available for {gender} students.")  
  
 else:  
 print(f"Cannot add {name}. No more seats available in bus.")  
  
 def show\_students(self):  
 print("\nStudents on the bus:")  
 for student in self.students:  
 print(f"Seat {student['seat\_number']}: {student['name']} ({student['gender']})")  
  
 def drop\_students(self):  
 print("\nDropping students at the department.")  
 self.students = []  
  
  
# Example usage  
bus = Bus()  
  
# Adding students to the bus  
bus.add\_student("Sara", "female")  
bus.add\_student("Ali", "male")  
bus.add\_student("Ahmad", "male")  
bus.add\_student("Zainab", "female")  
bus.add\_student("Maryam", "female")  
  
  
bus.show\_students()  
  
  
bus.drop\_students()  
  
  
bus.show\_students()

# OUTPUT:

# TASK 2:

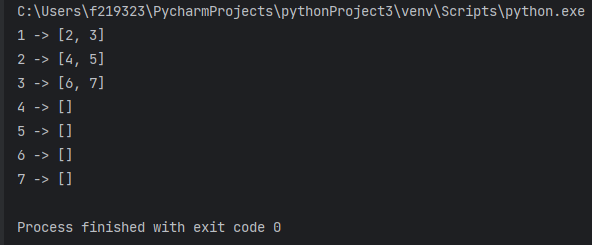
graph = {  
 'dunwich': {'blaxhall': 15, 'harwich': 53},  
 'blaxhall': {'dunwich': 15, 'harwich': 40, 'feering': 46},  
 'harwich': {'dunwich': 53, 'blaxhall': 40, 'clacton': 17},  
 'feering': {'blaxhall': 46, 'tiptree': 3, 'maldon': 11},  
 'tiptree': {'feering': 3, 'harwich': 31, 'clacton': 29, 'maldon': 8},  
 'clacton': {'harwich': 17, 'tiptree': 29, 'maldon': 40},  
 'maldon': {'feering': 11, 'tiptree': 8, 'clacton': 40}  
}  
  
for node, edges in graph.items():  
 print(f"{node} -> {edges}")

# OUTPUT:

# TASK 3:

graph = {  
 1: [2, 3],  
 2: [4, 5],  
 3: [6, 7],  
 4: [],  
 5: [],  
 6: [],  
 7: []  
}  
  
  
for node, children in graph.items():  
 print(f"{node} -> {children}")

# OUTPUT:



# TASK 4:

graph = {  
 'Arad': {'Zerind': 75, 'Sibiu': 140, 'Timisoara': 118},  
 'Oradea': {'Zerind': 71, 'Sibiu': 151},  
 'Zerind': {'Oradea': 71, 'Arad': 75},  
 'Timisoara': {'Arad': 118, 'Lugoj': 111},  
 'Lugoj': {'Mehadia': 70, 'Timisoara': 111},  
 'Mehadia': {'Drobeta': 75, 'Lugoj': 70},  
 'Drobeta': {'Mehadia': 75, 'Craiova': 120},  
 'Sibiu': {'Oradea': 151, 'Arad': 140, 'Fagaras': 99, 'Rimnicu Vilcea': 80},  
 'Rimnicu Vilcea': {'Sibiu': 80, 'Craiova': 146, 'Pitesti': 97},  
 'Craiova': {'Drobeta': 120, 'Pitesti': 138, 'Rimnicu Vilcea': 146},  
 'Fagaras': {'Sibiu': 99, 'Bucharest': 211},  
 'Pitesti': {'Craiova': 138, 'Rimnicu Vilcea': 97, 'Bucharest': 101},  
 'Neamt': {'Iasi': 87},  
 'Iasi': {'Neamt': 87, 'Vaslui': 92},  
 'Vaslui': {'Urziceni': 142, 'Iasi': 92},  
 'Urziceni': {'Bucharest': 85, 'Hirsova': 98, 'Vaslui': 142},  
 'Bucharest': {'Fagaras': 211, 'Giurgiu': 90, 'Pitesti': 101, 'Urziceni': 85},  
 'Giurgiu': {'Bucharest': 90},  
 'Hirsova': {'Eforie': 86, 'Urziceni': 98},  
 'Eforie': {'Hirsova': 86}  
}  
  
  
def dfs(graph, start, visited=None):  
 if visited is None:  
 visited = set()  
 print(start)  
 visited.add(start)  
 for neighbor in graph[start]:  
 if neighbor not in visited:  
 dfs(graph, neighbor, visited)  
  
# Starting from 'Arad'  
print("Traversal starting from 'Arad':")  
dfs(graph, 'Arad')  
  
# Starting from 'Fagaras'  
print("\nTraversal starting from 'Fagaras':")  
dfs(graph, 'Fagaras')

# OUTPUT:

