

# AI ASSISTED CODING 12.5

## LAB – 12.5

VIGHNESH BACHWAL

2303A51795

BATCH 12

### Task Description #1 (Sorting – Merge Sort Implementation)

- Task: Use AI to generate a Python program that implements the Merge Sort algorithm.

```
#use AI to generate the program of mergesort
def merge_sort(arr):
    if len(arr) > 1:
        mid = len(arr) // 2
        L = arr[:mid]
        R = arr[mid:]

        merge_sort(L)
        merge_sort(R)

        i = j = k = 0

        while i < len(L) and j < len(R):
            if L[i] < R[j]:
                arr[k] = L[i]
                i += 1
            else:
                arr[k] = R[j]
                j += 1
            k += 1

        while i < len(L):
            arr[k] = L[i]
            i += 1
            k += 1

        while j < len(R):
            arr[k] = R[j]
            j += 1
            k += 1

# Example usage:
arr = [38, 27, 43, 3, 9, 82, 10]
merge_sort(arr)
```

### OUTPUT

```
30           k += 1
31     # Example usage:
32     arr = [38, 27, 43, 3, 9, 82, 10]
33     merge_sort(arr)
34     print("Sorted array is:", arr) # Output: Sorted array is: [3, 9, 10, 27, 38, 43, 82]

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell +
```

PS C:\Users\ADMIN\AppData\Local\Programs\Microsoft VS Code> & C:\Users\ADMIN\AppData\Local\Python\pythoncore-3.14-64\python.exe "c:/Users/ADMIN/Desktop/AI\_AC LAB/1.py"
Sorted array is: [3, 9, 10, 27, 38, 43, 82]
PS C:\Users\ADMIN\AppData\Local\Programs\Microsoft VS Code> |

## Task Description #2 (Searching – Binary Search with AI Optimization)

- Task: Use AI to create a binary search function that finds a target element in a sorted list.

```
def binary_search(arr, target):
    left, right = 0, len(arr) - 1

    while left <= right:
        mid = left + (right - left) // 2

        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            left = mid + 1
        else:
            right = mid - 1

    return -1 # Target not found

# Example usage:
arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
target = 5
result = binary_search(arr, target)
if result != -1:
    print(f"Target {target} found at index: {result}") # Output: Target 5 found at index: 4
```

## OUTPUT

```
# Example usage:
arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
target = 5
result = binary_search(arr, target)
if result != -1:
    print(f"Target {target} found at index: {result}") # Output: Target 5 found at index: 4

VS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell +
```

C:\Users\ADMIN\AppData\Local\Programs\Microsoft VS Code> & C:\Users\ADMIN\AppData\Local\Python\pythoncore-3.14-64\python.exe "c:/Users/ADMIN/Desktop/AI\_AC LAB/binary\_search.py"

```
5 found at index: 4
```

## Task Description #3: Smart Healthcare Appointment Scheduling

**System** A healthcare platform maintains appointment records containing appointment ID, patient name, doctor name, appointment time, and consultation fee. The system needs to:

1. Search appointments using appointment ID.
2. Sort appointments based on time or consultation fee.

```

class Appointment:
    def __init__(self, appointment_id, patient_name, doctor_name, appointment_time, consultation_fee):
        self.appointment_id = appointment_id
        self.patient_name = patient_name
        self.doctor_name = doctor_name
        self.appointment_time = appointment_time
        self.consultation_fee = consultation_fee
class HealthcarePlatform:
    def __init__(self):
        self.appointments = []

    def add_appointment(self, appointment):
        self.appointments.append(appointment)

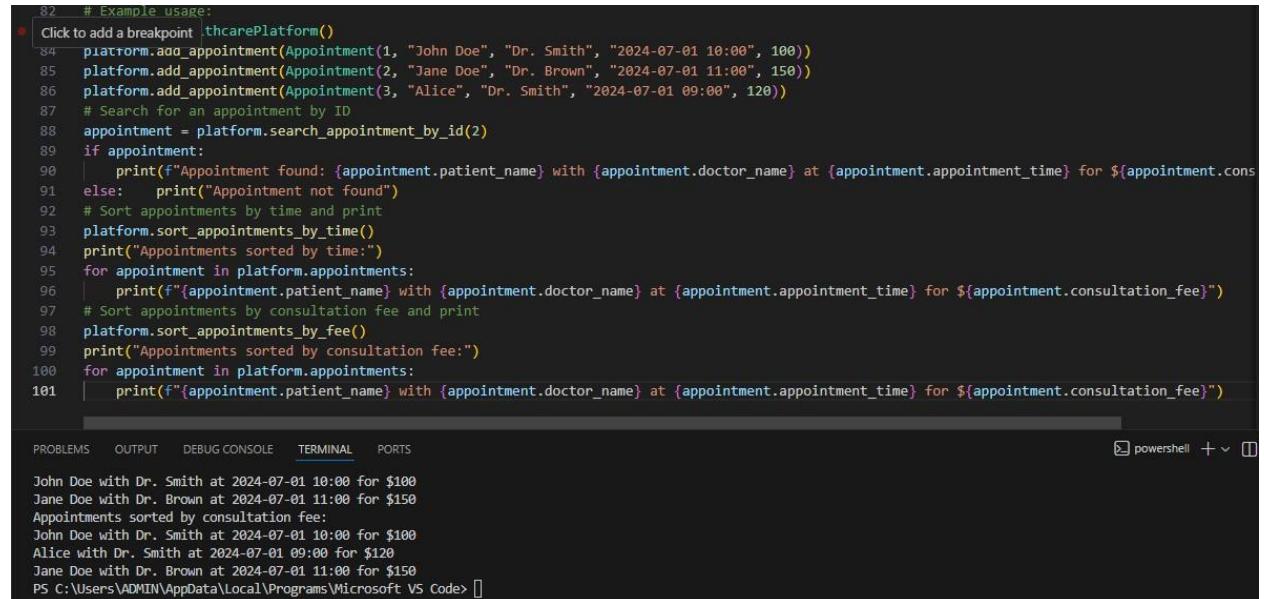
    def search_appointment_by_id(self, appointment_id):
        for appointment in self.appointments:
            if appointment.appointment_id == appointment_id:
                return appointment
        return None # Appointment not found

    def sort_appointments_by_time(self):
        self.appointments.sort(key=lambda x: x.appointment_time)

    def sort_appointments_by_fee(self):
        self.appointments.sort(key=lambda x: x.consultation_fee)
# Example usage:
platform = HealthcarePlatform()
platform.add_appointment(Appointment(1, "John Doe", "Dr. Smith", "2024-07-01 10:00", 100))
platform.add_appointment(Appointment(2, "Jane Doe", "Dr. Brown", "2024-07-01 11:00", 150))
platform.add_appointment(Appointment(3, "Alice", "Dr. Smith", "2024-07-01 09:00", 120))

```

## OUPUT



The screenshot shows the Microsoft VS Code interface with the terminal tab active. The terminal window displays the execution of a Python script that manages appointments in a healthcare platform. The output shows the addition of three appointments and their subsequent sorting by time and fee.

```

82 # Example usage:
83 Click to add a breakpoint:hcarePlatform()
84 platform.add_appointment(Appointment(1, "John Doe", "Dr. Smith", "2024-07-01 10:00", 100))
85 platform.add_appointment(Appointment(2, "Jane Doe", "Dr. Brown", "2024-07-01 11:00", 150))
86 platform.add_appointment(Appointment(3, "Alice", "Dr. Smith", "2024-07-01 09:00", 120))
87 # Search for an appointment by ID
88 appointment = platform.search_appointment_by_id(2)
89 if appointment:
90     print(f"Appointment found: {appointment.patient_name} with {appointment.doctor_name} at {appointment.appointment_time} for ${appointment.consultation_fee}")
91 else:
92     print("Appointment not found")
93 # Sort appointments by time and print
94 platform.sort_appointments_by_time()
95 print("Appointments sorted by time:")
96 for appointment in platform.appointments:
97     print(f"{appointment.patient_name} with {appointment.doctor_name} at {appointment.appointment_time} for ${appointment.consultation_fee}")
98 # Sort appointments by consultation fee and print
99 platform.sort_appointments_by_fee()
100 print("Appointments sorted by consultation fee:")
101 for appointment in platform.appointments:
102     print(f"{appointment.patient_name} with {appointment.doctor_name} at {appointment.appointment_time} for ${appointment.consultation_fee}")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
John Doe with Dr. Smith at 2024-07-01 10:00 for $100
Jane Doe with Dr. Brown at 2024-07-01 11:00 for $150
Appointments sorted by consultation fee:
John Doe with Dr. Smith at 2024-07-01 10:00 for $100
Alice with Dr. Smith at 2024-07-01 09:00 for $120
Jane Doe with Dr. Brown at 2024-07-01 11:00 for $150
PS C:\Users\ADMIN\AppData\Local\Programs\Microsoft VS Code> []

```

## Task Description #4: Railway Ticket Reservation System Scenario

A railway reservation system stores booking details such as ticket ID, passenger name, train number, seat number, and travel date. The system must:

1. Search tickets using ticket ID.

2. Sort bookings based on travel date or seat number

```
class Ticket:
    def __init__(self, ticket_id, passenger_name, train_number, seat_number, travel_date):
        self.ticket_id = ticket_id
        self.passenger_name = passenger_name
        self.train_number = train_number
        self.seat_number = seat_number
        self.travel_date = travel_date

class RailwayReservationSystem:
    def __init__(self):
        self.tickets = []

    def add_ticket(self, ticket):
        self.tickets.append(ticket)

    def search_ticket_by_id(self, ticket_id):
        for ticket in self.tickets:
            if ticket.ticket_id == ticket_id:
                return ticket
        return None # Ticket not found

    def sort_tickets_by_travel_date(self):
        self.tickets.sort(key=lambda x: x.travel_date)

    def sort_tickets_by_seat_number(self):
        self.tickets.sort(key=lambda x: x.seat_number)

# Example usage:
reservation_system = RailwayReservationSystem()
reservation_system.add_ticket(Ticket(1, "John Doe", "Train A", "A1", "2024-08-01"))
reservation_system.add_ticket(Ticket(2, "Jane Doe", "Train B", "B2", "2024-08-02"))
reservation_system.add_ticket(Ticket(3, "Alice", "Train A", "A2", "2024-08-01"))
# Search for a ticket by ID
```

## OUTPUT

```
134 # Search for a ticket by ID
135 ticket = reservation_system.search_ticket_by_id(2)
136 if ticket:
137     print(f"Ticket found: {ticket.passenger_name} on {ticket.train_number} seat {ticket.seat_number} for travel on {ticket.travel_date}")
138 else:
139     print("Ticket not found")
140 # Sort tickets by travel date and print
141 reservation_system.sort_tickets_by_travel_date()
142 print("Tickets sorted by travel date:")
143 for ticket in reservation_system.tickets:
144     print(f"{ticket.passenger_name} on {ticket.train_number} seat {ticket.seat_number} for travel on {ticket.travel_date}")
145 # Sort tickets by seat number and print
146 reservation_system.sort_tickets_by_seat_number()
147 print("Tickets sorted by seat number:")
148 for ticket in reservation_system.tickets:
149     print(f"{ticket.passenger_name} on {ticket.train_number} seat {ticket.seat_number} for travel on {ticket.travel_date}")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Tickets sorted by travel date:
John Doe on Train A seat A1 for travel on 2024-08-01
Alice on Train A seat A2 for travel on 2024-08-01
Jane Doe on Train B seat B2 for travel on 2024-08-02
Tickets sorted by seat number:
John Doe on Train A seat A1 for travel on 2024-08-01
Alice on Train A seat A2 for travel on 2024-08-01
Jane Doe on Train B seat B2 for travel on 2024-08-02
PS C:\Users\ADMIN\AppData\Local\Programs\Microsoft VS Code>
```

**Task Description #5: Smart Hostel Room Allocation System** A hostel management system stores student room allocation details including student ID, room number, floor, and allocation date. The system needs to:

1. Search allocation details using student ID.
2. Sort records based on room number or allocation date.

```

class RoomAllocation:
    def __init__(self, student_id, room_number, floor, allocation_date):
        self.student_id = student_id
        self.room_number = room_number
        self.floor = floor
        self.allocation_date = allocation_date
class HostelManagementSystem:
    def __init__(self):
        self.allocations = []

    def add_allocation(self, allocation):
        self.allocations.append(allocation)

    def search_allocation_by_student_id(self, student_id):
        for allocation in self.allocations:
            if allocation.student_id == student_id:
                return allocation
        return None # Allocation not found

    def sort_allocations_by_room_number(self):
        self.allocations.sort(key=lambda x: x.room_number)

    def sort_allocations_by_allocation_date(self):
        self.allocations.sort(key=lambda x: x.allocation_date)

# Example usage:
hostel_system = HostelManagementSystem()
hostel_system.add_allocation(RoomAllocation(1, "101", "1st Floor", "2024-09-01"))
hostel_system.add_allocation(RoomAllocation(2, "102", "1st Floor", "2024-09-02"))
hostel_system.add_allocation(RoomAllocation(3, "201", "2nd Floor", "2024-09-01"))
# Search for an allocation by student ID
allocation = hostel_system.search_allocation_by_student_id(2)

```

## OUTPUT

```

175 # Example usage:
176 hostel_system = HostelManagementSystem()
177 hostel_system.add_allocation(RoomAllocation(1, "101", "1st Floor", "2024-09-01"))
178 # (variable) hostel_system: HostelManagementSystem | "1", "1st Floor", "2024-09-02")
179 hostel_system.add_allocation(RoomAllocation(3, "201", "2nd Floor", "2024-09-01"))
180 # Search for an allocation by student ID
181 allocation = hostel_system.search_allocation_by_student_id(2)
182 if allocation:
183     print(f"Allocation found: Student ID {allocation.student_id} in room {allocation.room_number} on {allocation.allocation_date}")
184 else:
185     print("Allocation not found")
186 # Sort allocations by room number and print
187 hostel_system.sort_allocations_by_room_number()
188 print("Allocations sorted by room number:")
189 for allocation in hostel_system.allocations:
190     print(f"Student ID {allocation.student_id} in room {allocation.room_number} on {allocation.allocation_date}")
191 # Sort allocations by allocation date and print
192 hostel_system.sort_allocations_by_allocation_date()
193 print("Allocations sorted by allocation date:")
194 for allocation in hostel_system.allocations:
195     print(f"Student ID {allocation.student_id} in room {allocation.room_number} on {allocation.allocation_date}")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Allocations sorted by room number:
Student ID 1 in room 101 on 2024-09-01
Student ID 2 in room 102 on 2024-09-02
Student ID 3 in room 201 on 2024-09-01
Allocations sorted by allocation date:
Student ID 1 in room 101 on 2024-09-01
Student ID 3 in room 201 on 2024-09-01
Student ID 2 in room 102 on 2024-09-02
PS C:\Users\ADMIN\AppData\Local\Programs\Microsoft VS Code>

```

## Online Movie Streaming Platform

A streaming service maintains movie records with movie ID, title, genre, rating, and release year. The platform needs to:

1. Search movies by movie ID.
2. Sort movies based on rating or release year.

```
class Movie:  
    def __init__(self, movie_id, title, genre, rating, release_year):  
        self.movie_id = movie_id  
        self.title = title  
        self.genre = genre  
        self.rating = rating  
        self.release_year = release_year  
class MovieStreamingPlatform:  
    def __init__(self):  
        self.movies = []  
  
    def add_movie(self, movie):  
        self.movies.append(movie)  
  
    def search_movie_by_id(self, movie_id):  
        for movie in self.movies:  
            if movie.movie_id == movie_id:  
                return movie  
        return None # Movie not found  
  
    def sort_movies_by_rating(self):  
        self.movies.sort(key=lambda x: x.rating, reverse=True)  
  
    def sort_movies_by_release_year(self):  
        self.movies.sort(key=lambda x: x.release_year)  
# Example usage:  
platform = MovieStreamingPlatform()  
platform.add_movie(Movie(1, "Inception", "Sci-Fi", 8.8, 2010))  
platform.add_movie(Movie(2, "The Matrix", "Action", 8.7, 1999))  
platform.add_movie(Movie(3, "Interstellar", "Sci-Fi", 8.6, 2014))  
# Search for a movie by ID
```

## OUTPUT

```
222 # Example usage:  
223 platform = MovieStreamingPlatform()  
224 platform.add_movie(Movie(1, "Inception", "Sci-Fi", 8.8, 2010))  
225 platform.add_movie(Movie(2, "The Matrix", "Action", 8.7, 1999))  
226 platform.add_movie(Movie(3, "Interstellar", "Sci-Fi", 8.6, 2014))  
227 # Search for a movie by ID  
228 movie = platform.search_movie_by_id(2)  
229 if movie:  
230     print(f"Movie found: {movie.title} ({movie.genre}) with rating {movie.rating} released in {movie.release_year}")  
231 else:  
232     print("Movie not found")  
233 # Sort movies by rating and print  
234 platform.sort_movies_by_rating()  
235 print("Movies sorted by rating:")  
236 for movie in platform.movies:  
237     print(f"{movie.title} ({movie.genre}) with rating {movie.rating} released in {movie.release_year}")  
238 # Sort movies by release year and print  
239 platform.sort_movies_by_release_year()  
240 print("Movies sorted by release year:")  
241 for movie in platform.movies:  
242     print(f"{movie.title} ({movie.genre}) with rating {movie.rating} released in {movie.release_year}")  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell + ×
```

Movies sorted by rating:  
Inception (Sci-Fi) with rating 8.8 released in 2010  
The Matrix (Action) with rating 8.7 released in 1999  
Interstellar (Sci-Fi) with rating 8.6 released in 2014  
Movies sorted by release year:  
The Matrix (Action) with rating 8.7 released in 1999  
Inception (Sci-Fi) with rating 8.8 released in 2010  
Interstellar (Sci-Fi) with rating 8.6 released in 2014  
PS C:\Users\ADMIN\AppData\Local\Programs\Microsoft VS Code>

**Task Description #7: Smart Agriculture Crop Monitoring System** An agriculture monitoring system stores crop data with crop ID, crop name, soil moisture level, temperature, and yield estimate. Farmers need to:

1. Search crop details using crop ID.
2. Sort crops based on moisture level or yield estimate.

```
class Crop:  
    def __init__(self, crop_id, crop_name, soil_moisture_level, temperature, yield_estimate):  
        self.crop_id = crop_id  
        self.crop_name = crop_name  
        self.soil_moisture_level = soil_moisture_level  
        self.temperature = temperature  
        self.yield_estimate = yield_estimate  
class CropMonitoringSystem:  
    def __init__(self):  
        self.crops = []  
  
    def add_crop(self, crop):  
        self.crops.append(crop)  
  
    def search_crop_by_id(self, crop_id):  
        for crop in self.crops:  
            if crop.crop_id == crop_id:  
                return crop  
        return None # Crop not found  
  
    def sort_crops_by_moisture_level(self):  
        self.crops.sort(key=lambda x: x.soil_moisture_level)  
  
    def sort_crops_by_yield_estimate(self):  
        self.crops.sort(key=lambda x: x.yield_estimate, reverse=True)  
# Example usage:  
monitoring_system = CropMonitoringSystem()  
monitoring_system.add_crop(Crop(1, "Wheat", 30, 25, 1000))  
monitoring_system.add_crop(Crop(2, "Corn", 40, 28, 1500))  
monitoring_system.add_crop(Crop(3, "Rice", 35, 30, 1200))  
# Search for a crop by ID
```

## OUTPUT

The screenshot shows a code editor interface with a dark theme. On the left is the code editor pane containing Python code. On the right is the terminal pane showing the execution of the code and its output.

```
274 # Search for a crop by ID
275 crop = monitoring_system.search_crop_by_id(2)
276 if crop:
277     print(f"Crop found: {crop.crop_name} with soil moisture level {crop.soil_moisture_level}, temperature {crop.temperature}°C, and yield estimate {crop.yield_estimate} kg")
278 else:
279     print("Crop not found")
280 # Sort crops by moisture level and print
281 monitoring_system.sort_crops_by_moisture_level()
282 print("Crops sorted by soil moisture level:")
283 for crop in monitoring_system.crops:
284     print(f"{crop.crop_name} with soil moisture level {crop.soil_moisture_level}, temperature {crop.temperature}°C, and yield estimate {crop.yield_estimate} kg")
285 # Sort crops by yield estimate and print
286 monitoring_system.sort_crops_by_yield_estimate()
287 print("Crops sorted by yield estimate:")
288 for crop in monitoring_system.crops:
289     print(f"{crop.crop_name} with soil moisture level {crop.soil_moisture_level}, temperature {crop.temperature}°C, and yield estimate {crop.yield_estimate} kg")
```

TERMINAL

```
Crops sorted by soil moisture level:
Wheat with soil moisture level 30, temperature 25°C, and yield estimate 1000 kg
Rice with soil moisture level 35, temperature 30°C, and yield estimate 1200 kg
Corn with soil moisture level 40, temperature 28°C, and yield estimate 1500 kg
Crops sorted by yield estimate:
Corn with soil moisture level 40, temperature 28°C, and yield estimate 1500 kg
Rice with soil moisture level 35, temperature 30°C, and yield estimate 1200 kg
Wheat with soil moisture level 30, temperature 25°C, and yield estimate 1000 kg
PS C:\Users\ADMIN\AppData\Local\Programs\Microsoft VS Code>
```

**Task Description #8: Airport Flight Management System** An airport system stores flight information including flight ID, airline name, departure time, arrival time, and status. The system must:

1. Search flight details using flight ID.
2. Sort flights based on departure time or arrival time.

```

class Flight:
    def __init__(self, flight_id, airline_name, departure_time, arrival_time, status):
        self.flight_id = flight_id
        self.airline_name = airline_name
        self.departure_time = departure_time
        self.arrival_time = arrival_time
        self.status = status
class AirportFlightManagementSystem:
    def __init__(self):
        self.flights = []

    def add_flight(self, flight):
        self.flights.append(flight)

    def search_flight_by_id(self, flight_id):
        for flight in self.flights:
            if flight.flight_id == flight_id:
                return flight
        return None # Flight not found

    def sort_flights_by_departure_time(self):
        self.flights.sort(key=lambda x: x.departure_time)

    def sort_flights_by_arrival_time(self):
        self.flights.sort(key=lambda x: x.arrival_time)

# Example usage:
airport_system = AirportFlightManagementSystem()
airport_system.add_flight(Flight(1, "Airline A", "2024-10-01 08:00", "2024-10-01 10:00", "On Time"))
airport_system.add_flight(Flight(2, "Airline B", "2024-10-01 09:00", "2024-10-01 11:00", "Delayed"))
airport_system.add_flight(Flight(3, "Airline C", "2024-10-01 07:00", "2024-10-01 09:00", "On Time"))

```

## OUTPUT

```

316
317 # Example usage:
318 airport_system = AirportFlightManagementSystem()
319 airport_system.add_flight(Flight(1, "Airline A", "2024-10-01 08:00", "2024-10-01 10:00", "On Time"))
320 airport_system.add_flight(Flight(2, "Airline B", "2024-10-01 09:00", "2024-10-01 11:00", "Delayed"))
321 airport_system.add_flight(Flight(3, "Airline C", "2024-10-01 07:00", "2024-10-01 09:00", "On Time"))
322 # Search for a flight by ID
323 flight = airport_system.search_flight_by_id(2)
324 if flight:
325     print("Flight found: {} departing at {} arriving at {} with status {}".format(flight.airline_name, flight.departure_time, flight.arrival_time, flight.status))
326 else:
327     print("Flight not found")
328 # Sort flights by departure time and print
329 airport_system.sort_flights_by_departure_time()
330 print("Flights sorted by departure time:")
331 for flight in airport_system.flights:
332     print("{} departing at {} arriving at {} with status {}".format(flight.airline_name, flight.departure_time, flight.arrival_time, flight.status))
333 # Sort flights by arrival time and print
334 airport_system.sort_flights_by_arrival_time()
335 print("Flights sorted by arrival time:")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

powershell +

```

Flight found: Airline B departing at 2024-10-01 09:00 arriving at 2024-10-01 11:00 with status Delayed
Flights sorted by departure time:
Airline C departing at 2024-10-01 07:00 arriving at 2024-10-01 09:00 with status On Time
Airline A departing at 2024-10-01 08:00 arriving at 2024-10-01 10:00 with status On Time
Airline B departing at 2024-10-01 09:00 arriving at 2024-10-01 11:00 with status Delayed
Flights sorted by arrival time:
Airline C departing at 2024-10-01 07:00 arriving at 2024-10-01 09:00 with status On Time
Airline A departing at 2024-10-01 08:00 arriving at 2024-10-01 10:00 with status On Time
Airline B departing at 2024-10-01 09:00 arriving at 2024-10-01 11:00 with status Delayed

```