1. **Customer Transaction Analysis (Using List Pairs)**

Problem:

You are working for a retail store that tracks customer purchases. Each entry in the list contains a customer ID and the items they purchased. The prices for each item are also stored as a list of pairs (`[item, price]`). Your task is to:

- Find which customer spent the most money.
- Calculate the total money spent by each customer.
- Identify customers who made more than 5 purchases but spent less than $50.

Example Input:

```
transactions = [
    [1, ['milk', 'bread', 'butter']],
    [2, ['apple', 'milk']],
    [1, ['bread', 'orange juice']],
    [3, ['apple', 'milk', 'eggs']],
    [2, ['bread']],
    [3, ['bread', 'butter', 'coffee']],
]

prices = [
    ['milk', 1.5], ['bread', 2.0], ['butter', 3.0],
    ['apple', 1.0], ['orange juice', 4.5], ['eggs', 2.5], ['coffee', 5.0]
]
```

Output:

- Customer with the highest total spend.
- Total spend per customer.
- Customers who made more than 5 purchases but spent less than $50.

---

2. **Event Planning for a Conference (Using List Pairs)**

Problem:

You are organizing a conference where attendees sign up for various sessions. Each session has a limited capacity, and you need to manage who gets to attend each session and who is placed on the waiting list. Your input is a list where each entry contains the attendee's name and their preferred sessions. The session capacities are represented as a list of pairs (`[session, capacity]`). Your tasks:

- Assign attendees to sessions without exceeding the capacity.
- Generate waiting lists for full sessions.
- Identify attendees who couldn't be assigned to their preferred sessions.

Example Input:

```
attendees = [
    ['Alice', ['Session1', 'Session3']],
    ['Bob', ['Session2']],
    ['Charlie', ['Session1']],
    ['Diana', ['Session1', 'Session2', 'Session3']],
    ['Edward', ['Session1', 'Session2']],
    ['Fiona', ['Session3']],
]

session_capacities = [
    ['Session1', 2], ['Session2', 3], ['Session3', 1]
]
```

Output:

- Assigned attendees for each session.
- Waiting lists for full sessions.
- Unassigned attendees.

---

3. **Flight Booking System (Using List Pairs)**

Problem:

You are developing a flight booking system. Each reservation contains the passenger's name, the flight they wish to book, and their preferred seat type (window, aisle, or middle). Flights have limited numbers of each seat type, and these are stored as list pairs (`[seat_type, available_count]`). Your tasks are:

- Assign passengers to their preferred seats, or assign them to any available seat if their preference isn't available.
- Track the number of unfulfilled seat preferences.
- Calculate the seat utilization rate for each flight.

Example Input:

```
reservations = [
    ['John', 'Flight1', 'window'],
    ['Jane', 'Flight1', 'aisle'],
    ['Mike', 'Flight2', 'window'],
    ['Sara', 'Flight1', 'window'],
    ['Tom', 'Flight2', 'aisle'],
]

flights = [
    ['Flight1', [['window', 2], ['aisle', 3], ['middle', 5]]],
    ['Flight2', [['window', 1], ['aisle', 2], ['middle', 4]]],
]
```

Output:

- Assigned seats for each passenger.
- Unfulfilled seat preferences.
- Seat utilization rate for each flight.

---

4. **Delivery Route Optimization (Using List Pairs)**

Problem:

You are managing a delivery service. The deliveries list contains packages, where each entry has a package ID and the distance to the delivery destination (in kilometers). Each driver has a maximum daily range they can cover. The list `drivers` contains pairs of driver names and the maximum distance they can cover. Your tasks:

- Assign packages to drivers so they can cover all deliveries without exceeding their range.
- Identify undelivered packages (if no drivers can deliver them).
- Calculate the total distance each driver needs to cover.

Example Input:

```
deliveries = [
    ['Package1', 15],  # 15 km
    ['Package2', 25],
    ['Package3', 10],
    ['Package4', 40],
    ['Package5', 30]
]

drivers = [
    ['Driver1', 60],  # can cover 60 km
    ['Driver2', 50],
    ['Driver3', 20]
]
```

Output:

- Assignment of packages to drivers.
- List of undelivered packages.
- Total distance covered by each driver.

---

5. **Movie Recommendation System (Using List Pairs)**

Problem:

You are building a movie recommendation system. Each entry in the list represents a user, and for each user, there is a list of the movies they have watched along with their ratings (stored as pairs). Your tasks:

- Recommend movies to each user based on the preferences of users who have watched similar movies.
- Identify the top 5 recommended movies for each user.
- Find users who have watched fewer than 5 movies.

Example Input:

```
users = [
    ['User1', [['MovieA', 5], ['MovieB', 4]]],
    ['User2', [['MovieA', 4], ['MovieC', 5], ['MovieB', 3]]],
    ['User3', [['MovieC', 5], ['MovieD', 2]]],
    ['User4', [['MovieE', 3]]],
]
```

Output:

- Movie recommendations for each user.
- Top 5 recommended movies for each user.
- Users with fewer than 5 watched movies.

---

6. **Warehouse Inventory Management (Using List Pairs)**

Problem:

You are managing a warehouse, and each product is represented by a list entry containing the product's ID, its quantity, and a list of suppliers (with each supplier's name and their price for the product). Your tasks:

- Restock products that fall below a certain threshold by reordering from the supplier offering the lowest price.
- Identify products that are out of stock and have no suppliers.
- Sort the products based on total quantity available.

Example Input:

```
inventory = [
    ['Product1', 50, [['SupplierA', 5], ['SupplierB', 6]]],
    ['Product2', 0, [['SupplierC', 7]]],
    ['Product3', 20, []],
    ['Product4', 10, [['SupplierA', 5]]],
]

reorder_threshold = 15
```

Output:

- Restocked products and their suppliers.
- Out-of-stock products with no suppliers.
- Products sorted by quantity.

---

7. **University Course Scheduling (Using List Pairs)**

Problem:

You are tasked with assigning students to courses. Each student has a list of their preferred courses. Each course has a maximum capacity and a scheduled time. These are represented as list pairs (`[course, capacity]`, `[course, time]`). Your tasks:

- Assign students to courses based on their preferences, ensuring no schedule conflicts.
- Maximize attendance while adhering to course capacities.
- Identify students who couldn't be assigned to their preferred courses.

Example Input:

```
students = [
    ['Student1', ['CourseA', 'CourseB']],
    ['Student2', ['CourseC', 'CourseB']],
    ['Student3', ['CourseA']],
    ['Student4', ['CourseB', 'CourseC']],
]

courses = [
    ['CourseA', ['capacity', 2], ['time', '10:00 AM']],
    ['CourseB', ['capacity', 3], ['time', '11:00 AM']],
    ['CourseC', ['capacity', 2], ['time', '10:00 AM']]
]
```

Output:

- Assigned students for each course.
- Students who couldn't be assigned to any of their preferred courses.
- Conflicts resolved by adjusting schedules.
