EE/EC3202: Data Structures and Algorithms Project Instructions (2024)

As part of this module, you are required to create a project that demonstrates your understanding of the data structures and algorithms covered in the course, alongside the application of Object-Oriented Programming (OOP) concepts. Below are the detailed instructions for the project:

General Guidelines

1. **Project Theme:**
    - Choose a theme for your project as a group. Example project ideas include (Must not limited to these ideas; think your own):
        - Library Book Management System
        - University Cafeteria Management System
        - Student Gradebook
        - Inventory Management System
        - Music Playlist Manager
    - Unique ideas are encouraged. If selecting a similar theme as another group, ensure your approach is distinct (e.g., different use cases, workflows, or structure).
2. **Group Composition:**
    - Each group should have **three members**.
    - **One group** may consist of **four members** (considering the total number of students) if the project has an enhanced scope.
    - Choose your group members independently.
3. **Language & Tools:**
    - Implement the project in **C#**.
    - You **cannot** use built-in data structures or algorithms from C#. Instead, use the data structures and algorithms implemented during the course. These may be imported as libraries.
4. **OOP Concepts:**
    - Apply OOP principles effectively, including:
        - **Abstraction**
        - **Encapsulation**
        - **Inheritance**
        - **Polymorphism**

5. **Creativity & Depth:**
    - Use a variety of data structures relevant to the project requirements.
    - Be innovative and thoughtful in your implementation.
    - **Bonus marks will be awarded to groups with innovative ideas or project depth.**

Algorithm Implementation & Analysis

1. **Team Collaboration:**
    - Divide algorithmic tasks among team members. Each member should:
        - Implement and test a different algorithm for the same tasks.
        - Example: In a library management system: sort books by name or ISBN (there can be many other tasks...)
            - Member 1: Use **Bubble Sort**
            - Member 2: Use **Merge Sort**.
            - Member 3: Use **Quick Sort**.
2. **Performance Analysis:**
    - Compare the performance of different algorithms using methods such as:
        - **Execution time analysis**
        - **Theoretical analysis (e.g., Big O notation)**
    - Identify the most efficient algorithm based on observations.
    - Document all results in your project report and presentation. Include visualizations (e.g., graphs or charts) to support your analysis.

Project Proposal Submission

- Submit your project proposal using the provided Google Sheet. Ensure all required fields are completed.
- **Proposal Submission Link:**
  EE/EC3202 Data Structures and Algorithms 2024: Project Proposals
- The proposal submission deadline will be announced on ELMS. Be sure to submit by the specified date.

For example, you can go through the following project scope.

Airline Reservation System

- **Description:** Build a robust system for managing airline bookings with advanced features for searching, booking, cancellation, and flight management. The system should be scalable to handle thousands of flights and passengers efficiently.
  - **Requirements**
    1. **Flight Management**
       - **a.** Manage a database of flights, including flight numbers, departure and arrival times, destinations, durations, and seat capacities.
       - **b.** Enable adding, updating, and deleting flight details dynamically.
       - **c.** Support connecting flights and provide the shortest route suggestions between two cities.
    2. **Booking System**
       - **a.** Allow users to search for flights by source, destination, date, and time.
       - **b.** Implement seat allocation, supporting different seat classes (economy, business, first class).
       - **c.** Track passenger details, including name, ID, and contact information.
       - **d.** Automatically assign seats based on preferences (e.g., window, aisle).
       - **e.** Handle group bookings and ensure they are seated together when possible.
    3. **Dynamic Pricing**
       - **a.** Implement real-time dynamic pricing based on demand, booking time, and seat availability.
       - **b.** Offer discounted fares for early bookings and premium pricing for last-minute bookings.
    4. **Waiting List Management**
       - **a.** Manage waiting lists for fully booked flights.
       - **b.** Notify passengers when a seat becomes available, and update the list automatically.

5. **Flight Rescheduling and Cancellations**
    - **a.** Allow passengers to reschedule their bookings without losing seat preferences.
    - **b.** Handle cancellations, refunds, and reallocation of seats efficiently.

- **Suggested Data Structures:**
    1. **Graphs**: Represent flight networks to handle connecting flights and calculate the shortest path between cities (e.g., Dijkstra's).
    2. **Trees**: Optimize searches for flight codes or cities.
    3. **Priority Queues or Heaps**: Dynamically allocate seats based on preferences.
    4. **Linked Lists**: Manage the waiting list for efficient updates and notifications.
