FLIGHT RESERVATION & SEATING MANAGEMENT SYSTEM USING DATA STRUCTURES AND ALGORITHMS

(Working Duration: Monday of Week 5 to Monday of Week 10 – 30 Marks)

Background:

Modern airlines rely heavily on efficient digital reservation systems to manage passenger bookings, seating allocation, and flight manifests. These systems ensure that airlines can process large volumes of reservations, minimize errors, and optimize seat usage.

A well-designed flight reservation system not only stores and manages passenger details, but also handles cancellations, waitlists, seat lookups, and real-time updates to flight status. Data structures such as **arrays** and **linked lists** play a crucial role in determining the system's efficiency.

An array-based structure enables fast, index-based seat access, while a linked list-based structure offers flexibility for dynamic insertion and deletion of passenger records. Evaluating and comparing these approaches provides insight into performance trade-offs in real-world reservation systems.

In this assignment, your team will design and develop a **Flight Reservation & Seating Management Component** using **Array** and **Linked List** data structures, as well as **appropriate searching and traversal algorithms**. You will also perform a performance evaluation focusing on **time and memory efficiency**.

Question:

You are given a dataset:

- **flight_passenger_data.csv** — contains passenger booking data (PassengerID, Name, SeatRow, SeatColumn, Class)

Using the dataset above, your group must design and implement two versions of a flight reservation component:

Technical Requirements List:

Your solution must include:

1. Data Structure Components

- **Array-Based Component:**
  Store and manage seat assignments using 2D arrays and maintain passenger lists using 1D arrays.
- **Linked List-Based Component:**
  Use a singly or doubly linked list to store passenger records dynamically.

2. Core Reservation Functionalities

Each version must implement:

- **Reservation (Insertion):** Allocate an available seat to a new passenger.
- **Cancellation (Deletion):** Remove a passenger using PassengerID and free the seat.
- **Seat Lookup (Search):** Retrieve passenger details based on PassengerID.
- **Manifest & Seat Report:** Print full seating chart and passenger manifest.

3. Driver Program

A main program that:

- Loads initial dataset from file
- Executes all reservation functionalities
- Displays the seating grid and manifest

4. Performance Evaluation

Compare:

- Time complexity (Big O)
- Memory usage trade-offs
  for the **Array-based** vs **Linked List-based** implementations.

Task Distribution:

Design of Data Structure	Distribution	Requirement
Array-Based Component	Team Member 1 Team Member 2	Refer to technical requirements list
Linked List-Based Component	Team Member 3 Team Member 4	Refer to technical requirements list

Minimum Requirements for Lab Work #1

1. Each group may have up to **FOUR (4) members**.

2. This assignment requires the development of both:

   a) Array Implementation

   - Implement 2D and/or 1D arrays to store passenger data.
   - Support searching, seat traversal, and seat allocation.
   - Allow basic sorting/filtering operations (e.g., list passengers by row, class).

   b) Linked List Implementation

- Implement a singly or doubly linked list.
- Provide node-based insertion, deletion, searching, and traversal.
- Support similar operations as the array version.

3. **Performance Comparison:**
   Provide analysis of:

   - Time Efficiency: Static (arrays) vs dynamic (linked list) - Search, insertion, deletion
   - Space Efficiency: Static (arrays) vs dynamic (linked list)

4. **Error Handling:**
   - Implement **robust error handling and data validation** to address unexpected formats or missing data.

Submission Guidelines #1: Program and Video Submission – Lab Work #1 (15 Marks)

1. **Programming Language:** Use C++ to develop both programs.

2. **No built-in containers** like `<vector>` or `<list>`, you must create your own data containers with the basic operations.

3. **ZIP Submission:**
   - Include only `.cpp`, `.hpp`, and `.csv`/text files.
   - Follow this file naming format: `<GroupNo>_<TeamLeaderID>_<1stMemberID>_<2ndMemberID>_<3rdMemberID>.zip`
   - For example, "`G1_TP012345_TP014556_TP067554_TP034325.zip`"

4. **Video Recording:**
   - Each team member must also upload **ONE (1)** video recording and the maximum video duration is **5 minutes**.
   - Each member must relate his/her explanation to the workload matrix distribution table provided in the Word document.
   - Compress the final video recording to **under 200MB** before submitting.
   - If the video exceeds the specified time limit (5 minutes per member), it will only be assessed up to the specified duration and will affect the mark.
   - Videos must be recorded at normal speed (1x) and cannot be sped up or adjusted to meet the demo video duration requirements.
   - The video recording file must adhere to the following name format: `<GroupNo>_<MemberID>.mp4`
   - For example, "`G1_TP012345.mp4`"

5. Refer to **Page 7** for marking criteria of this Lab Evaluation Work #1 submission.

Submission Guidelines #2: Documentation Submission – Solution Work (15 Marks)

What to include in your documentation:

1. **Cover Page**

2. **Workload Matrix Table with Signature**
   Note that this table will impact each member's personal final mark in Lab Evaluation Work #1 based on their stated contribution percentage.

3. **Theoretical Explanation**
   - Explain data containers and operations' implementations in your chosen data structures and algorithm(s).

4. **Input-Output Screenshots**
   - Include system input and output screenshots.

5. **Summary Discussions**
   - Analyze system efficiency **(execution time, time efficiency, space efficiency, or etc.)**
   - Summarize and discuss observations made during development.
   - Critically evaluate strengths and weaknesses of the code.

6. **Conclusion and Reflection**
   - Summarize key findings.
   - Discuss potential improvements and system weaknesses.
   - Share personal thoughts on the assignment.

7. **References (if applicable)**

   - Properly cite external sources (APA format).

   Failure to reference code will be treated as plagiarism.

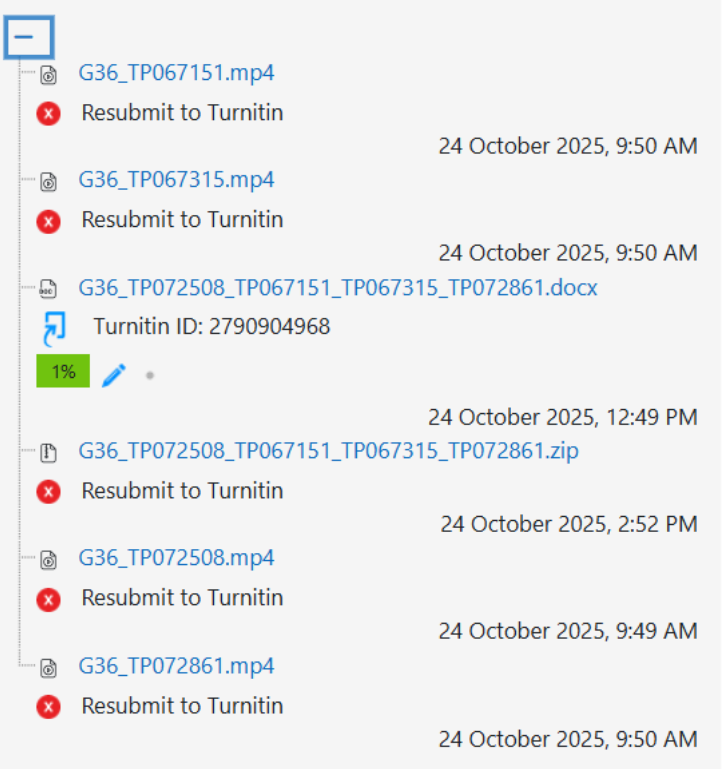
8. **Appendix (if applicable)**

Submission & Formatting:

- File Naming Format:
  `<GroupNo>_<TeamLeaderID>_<Member1ID>_<Member2ID>_<Member3ID>.docx`
- For example, `G1_TP012345_TP014556_TP067554_TP034325.docx`

Deadline:	**Refer to the Moodle**
Max Pages:	**30 pages**
Max Words:	**4500 words**

Sample Moodle Submission



MARKING CRITERIA

(Lab Evaluation Work #1 - 15 MARKS)

This Lab Evaluation Work #1 will be evaluated according to the following performance criteria:

Assessment Components	Inclusive	15 Marks
CLO3: Lab Evaluation Task #1 – 5-Minute Video Recording(Individual) ***(Assessment will be based on individual performance)***		
Practical Skills: Use of Data Structures & Algorithms + Personal Understanding		
Utilization of data structures	Technical Proficiency	
Implementation of relevant algorithms	Technical Proficiency	
Demonstrates understanding of data structures/algorithms used	Comprehensive Understanding	
Justifies choices of structures/algorithms	Insightful Justification	
Video Recording	Submission of video Recording	

MARKING CRITERIA

(Solution Work - 15 MARKS)

This solution work will be evaluated according to the following performance criteria:

Assessment Components	Inclusive	15 Marks
CLO2: Solution Work - Documentation ***(Assessment will be based on group component)***		
Theoretical Explanation (e.g., Data Structures, Algorithms)	Clear explanation	
Input Output Screenshots	Adequate Screenshots	
Summary Discussions (Inclusive Time and/or Space Complexity)	Clear and insightful analysis	
Conclusion & Reflection (Other Relevant / Importance Discussions)	Clearly highlighted and insightful	
Content Organization	Well-structured and logical flow	

Approximation of Total Pages for the documentation: **30 (max).**
Approximation of Words for the documentation: **4500 words (max)**