ANALYSING FINANCIAL TRANSACTIONS FOR FRAUD DETECTION USING DATA STRUCTURES AND ALGORITHMS

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

Background:

Recently, a significant financial fraud case came to light involving unauthorized transactions from multiple customer accounts at a mid-sized digital bank. The fraudulent activity went undetected for several hours, during which over $2 million was siphoned off. The incident has prompted regulatory scrutiny and raised concerns about the robustness of security infrastructure in online banking platforms, emphasizing the urgent need for more advanced fraud detection systems and better customer awareness.

One of the key challenges in financial institutions is **efficiently managing and processing vast amounts of financial transaction data**. The ability to **search for specific transactions and sort large datasets efficiently** is crucial for optimizing business operations.

Data structures play a vital role in these processes. **Arrays and linked lists** are fundamental data structures that can be used to store and manage financial data, while different **searching and sorting algorithms** determine the efficiency of retrieving and analyzing information.

In this assignment, your goal is to evaluate how different search and sorting algorithms perform when organising financial transaction data and explore how the choice of data structure, such as array or linked list, affects performance in terms of time and memory efficiency.

Question:

In this assignment, you are provided with **a dataset**:

- **financial_fraud_detection.csv** – Contains generated financial transactions to simulate real-world behavior for fraud detection research. (transaction details, behavioral features and metadata, fraud indicators).

You are required to develop data collection classes using array-based and linked based to store the financial transactions. Following that, appropriate search and sort algorithms are to implement for the following functional requirements, but not limited to, including display the financial transaction on the console output.

1. Store financial transactions separately according to payment channel
2. Sort financial transactions by location in ascending order
   *(**Note:** Select an efficient sorting algorithm according to the data structure)*
3. Search the specific transaction type
   *(**Note:** Select an efficient searching algorithm according to the data structure)*
4. Generate the JSON format from the specific data collection
   *(**Hint:** To export data to **JSON** using **C++**, you could use a third-party library like **nlohmann/json**)*
5. The MAIN PROGRAM to drive the functionalities for **1)** to **4)** above

**** Any logical assumption for the functional requirements may be accepted***

Minimum Requirements for Lab Work #1

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

2. This assignment requires the development of either one of two or both:

   a) Array Implementation

   a) Implement an **array-based** data structure to store and process the financial transactions.
   b) Perform operations like sorting, searching, and filtering transactions.

   b) Linked List Implementation

   a) Implement a **linked list-based** data structure to store the same data.
   b) Similar to the array-based approach, allow efficient operations like sorting and searching.

3. **Sorting Requirement:**
   - Sort financial transactions by **location** using suitable **sorting algorithm** for the data structures.

4. **Search Requirement:**
   - Implement **search algorithms** to filter and retrieve relevant transactions based on specific conditions.

5. **Performance Comparison:**
   - Compare the time and memory efficiency of search and sorting algorithms for **Arrays vs Linked Lists**.

6. **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 structures.

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 leader must also upload **ONE (1)** video recording and the maximum video duration is **30 minutes**.
   - All members must participate (each member speaks maximum of 5 minutes).
   - Each member must relate their 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 (30 minutes), it will only be assessed up to the specified duration.
   - Videos must be recorded at normal speed (1x) and cannot be speed up or adjusted to meet the demo video duration requirements.
   - The video recording file must adhere to the following name format: `<GroupNo>_<TeamLeaderID>_<1stMemberID>_<2ndMemberID>_<3rdMemberID>.mp4`
   - For example, "`G1_TP012345_TP014556_TP067554_TP034325.mp4`"

5. Refer to the **Page 5** 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 manipulations in your chosen data structures.

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

5. **Summary Discussions**
   - Analyze system efficiency **(execution time, time efficiency, space efficiency, 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**

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 – 30-Minute Video Recording ***(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	

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 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)**