Term Project: City RideShare Trip Analyzer

CMP2003 Data Structures and Algorithms (C++)

Fall 2025

General Rules & Guidelines

All students must adhere to the following rules for the Term Project. Failure to comply may result in grade reduction or disqualification.

1. **Group Composition:**
   - Projects must be completed in groups of **minimum 2, maximum 3** students.
   - The entire application must be implemented by the group members.
2. **Library Restrictions:**
   - You are restricted to **Standard Template Library (STL)** components only.
   - Third-party libraries are strictly prohibited.
   - If you require a data structure or container not present in the STL, you **must implement it yourself** from scratch.
3. **Late Submission Policy:**
   - Late submissions are accepted for up to **3 days** after the deadline.
   - A penalty of **-10 points** will be applied for every day of delay.
   - Submissions later than 3 days will not be graded.
4. **AI Usage & Oral Defense Policy:**
   - **AI Detection:** If the use of AI tools (ChatGPT, GitHub Copilot, etc.) is detected in your code or report, the group members will be immediately summoned for an oral representation.
   - **Zero Tolerance:** If group members fail to answer questions regarding their own code during this defense, they will automatically receive a grade of **0** for the project.
   - **Random Checks:** The instructor reserves the right to select **random teams** for oral representation, regardless of AI suspicion, to verify authorship and understanding.
5. **Test Data:**
   - Public sample data is provided for development.
   - Grading will be performed using a **Hidden Large-Scale Dataset** (hundreds of thousands of records).

1 Project Overview

Title: Urban Mobility Analytics – High-Volume Trip Log Processor
Context: The Urban Mobility Analytics Department requires a high-performance analysis tool to process massive volumes of ride-share log data. The current infrastructure cannot handle the growing dataset size efficiently.

Objective: Your goal is to design a C++ application that parses, aggregates, and analyzes trip logs to identify high-density traffic zones and peak operational hours. This project focuses strictly on **correctness under constraint** and **algorithmic efficiency**. You must select appropriate data structures to handle millions of records within strict time limits.

2 Technical Specifications

2.1 Input Data

The application must process a CSV file named `Trips.csv`. Each line represents a single trip with the following schema:

`TripID, PickupZoneID, DropoffZoneID, PickupDateTime, DistanceKm, FareAmount`

Example Record:

```
1000001, ZONE034, ZONE102, 2023-01-10 08:42, 7.4, 82.5
```

2.2 Core Functional Requirements

1. **Data Ingestion & Parsing:** Read the CSV file line-by-line. Parse the `PickupDateTime` to extract the hour (integer 0-23).
2. **Aggregation:**
   - Calculate total trips for every unique `PickupZoneID`.
   - Calculate total trips for every unique combination of `(PickupZoneID, Hour)`.
3. **Reporting:**
   - Display **Top 10 Pickup Zones** sorted by total trip count (descending).
   - Display **Top 10 Busy Slots** (PickupZoneID + Hour) sorted by trip count (descending).
4. **Performance Metrics:** Display the total execution time of the analysis.

3 Detailed Test Scenarios (Correctness: 70%)

To ensure robustness suitable for a Data Structures curriculum, your code will be tested against the following scenarios. Each test carries a specific weight towards the correctness grade.

Test Category A: Data Integrity & Edge Cases (15% Total)

- **The "Dirty Data" Test (5%):** Input files will contain lines with missing columns, non-numeric IDs, or invalid timestamps. Your program **must skip** these without crashing.
- **The Empty File Test (5%):** If the input file is empty, the program must terminate gracefully (no segmentation faults) and provide an empty output or message.
- **Boundary Hours (5%):** Trips occurring exactly at `00:00` and `23:59` must be correctly categorized into Hour 0 and Hour 23.

Test Category B: Algorithmic Logic & Sorting (20% Total)

- **The "Tie-Breaker" Test (10%):** If two zones have the exact same trip count, you must use a deterministic secondary sort key (e.g., alphanumerical order of `PickupZoneID`) to ensure consistent output.
- **The "Single Hit" Test (5%):** A dataset where every zone appears exactly once. The "Top 10" must be the first 10 zones sorted by ID.
- **Case Sensitivity (5%):** `zone01` and `ZONE01` must be treated as distinct IDs.

Test Category C: Stress & Scalability (35% Total)

- **The "High Collision" Test (10%):** 1,000,000 records belonging to only **2** unique zones. Tests if your implementation handles high frequency counts efficiently.
- **The "High Cardinality" Test (10%):** 100,000 records where **every** record has a unique ID. Tests memory overhead and storage efficiency.
- **The Volume Test (15%):** A hidden dataset with 5,000,000+ rows. Inefficient implementations will time out. The program must produce the correct output within the maximum allowed time to receive these points.

4 Performance Requirements (Performance: 20%)

Your submission will be ranked against other groups based on execution time on the **Volume Test**.

- **Target:** Process massive files within a reasonable timeframe.
- **Ranking:** The fastest correct implementations receive full points; others are scaled linearly.

5 Report & Documentation (Report: 10%)

A PDF report is mandatory and must include:

1. **Design Justification:** Explicitly state *why* you chose your specific data structures and algorithms over other potential alternatives.
2. **Complexity Analysis:** Provide Big-O time complexity for Insert, Search, and Sort operations.
3. **Tie-Breaking Strategy:** Explain how your code handles ties in the Top 10 lists.

6 Evaluation Criteria Summary

Final Deliverable: Single ZIP file containing source code (.cpp, .h) and the PDF Report. Do NOT include datasets.

Component	Weight	Description
Correctness	**70%**	Sum of weighted scores from Test Categories A, B, and C on the Hidden Large Scale Dataset.
Performance	**20%**	Execution speed ranking compared to other groups on the hidden dataset.
Report	**10%**	Clarity of design, Big-O analysis, and explanation of sorting strategies.