**Algorithms and Data Structure**  
**Artifact Title:** CS-210 Grocery Tracker (Inventory System Using Maps)  
**Creation Date:** Originally developed in November 2024 for CS-210: Programming Languages  
**Enhancement Date:** July 2025

**Artifact Description**  
This artifact is a C++ console application developed in CS-210, designed to manage a grocery inventory system using the std::map data structure. The application loads item data from a file, tracks quantities, and provides an interactive menu for managing the inventory. It demonstrates the use of maps for efficient key-value storage and retrieval.

**Justification for Inclusion**  
I selected this artifact because it reflects my understanding of core algorithmic logic and data structure design in C++. It highlights my ability to use associative containers for real-world data organization and manipulation. This project showcases my skills in:

* File parsing and data persistence.
* Use of std::map for inventory tracking.
* User-driven menu interactions.
* Algorithmic thinking in update, search, and display operations.

**Enhancements Performed**  
Following the code review in Module One, I implemented the following enhancements:

* Updated file parsing to read from and save to a key value format (item quantity) rather than counting repeated lines.
* Implemented item addition, removal, and quantity update features through a menu-driven interface.
* Ensured changes persist across sessions by writing back to the same file using a relative path.
* Refactored the code to be more modular, maintainable, and portable across systems.

These changes improved the program’s usability, file structure consistency, and persistent data handling.

**Course Outcomes Addressed**  
This enhancement contributes to the following program outcomes:

* **Algorithms and Data Structures:** Utilized std::map for constant-time lookup and implemented algorithms to manipulate item quantities.
* **Problem Solving:** Redesigned file reading/writing logic to support real-time updates and persistent storage in a format that enables future loading.
* **Software Design:** Applied modular design principles by separating responsibilities (menu interaction, data handling, and file I/O).

**Reflection**  
Enhancing this artifact taught me the importance of choosing the right data format and structure when working with persistent data. A major challenge was ensuring that file I/O worked seamlessly using a relative path so the program would remain portable. I also had to adapt the loading logic to parse structured lines correctly and maintain consistent formatting between sessions.  
This process deepened my understanding of file parsing in C++, algorithmic logic for managing dynamic data, and how to build reliable, user-facing software features that reflect back-end changes.

**Conclusion**  
This artifact and its enhancements demonstrate my growing proficiency in designing and implementing algorithmic solutions using structured data. It reflects my ability to build, refactor, and extend C++ programs in a way that emphasizes both functionality and maintainability. This enhancement marks meaningful progress toward competency in algorithms and data structure management.