**Victor Udeh CS 300 WK2  
Code Reflection**

Purpose of the Code

The VectorSorting.cpp program is designed to demonstrate the use of vector data structures in C++ and the implementation of sorting algorithms, specifically selection sort and quick sort. The primary functionalities include:

1. Reading Bid Data from a CSV File: It utilizes a csv::Parser to read and parse bid data from a CSV file, showcasing file handling and parsing in C++.
2. Data Storage and Management: The program stores bid data in a vector of Bid structures. This demonstrates effective use of standard library containers for managing a collection of custom objects.
3. Sorting Algorithms: Two sorting algorithms are implemented - selection sort and quick sort - to order bids based on their titles. This part of the code serves an educational purpose in comparing these algorithms in terms of implementation and performance.
4. Performance Measurement: Execution time for each sorting algorithm is measured and output to the console, providing a practical demonstration of algorithm efficiency.

Development Experience

Developing this program was both challenging and enlightening. Key experiences include:

1. CSV Parsing: Integrating and using the csv::Parser required understanding external library usage in C++. This part emphasized the importance of handling external dependencies.
2. Algorithm Implementation: Coding the sorting algorithms was a crucial exercise in understanding algorithm logic and translating it into efficient C++ code.
3. Debugging: Encountering and resolving various bugs, especially in sorting algorithms, honed debugging skills. It involved careful examination of loops and conditions to ensure correctness.
4. Performance Analysis: Implementing performance measurement provided insights into how different algorithms perform under the same data conditions.

Challenges and Solutions

* Linker Errors: Initially faced linker errors due to missing implementations of the csv::Parser. This was resolved by ensuring all relevant source files were correctly included and linked.
* Sorting Logic: Debugging the sorting algorithms, particularly the quick sort partition logic, required systematic testing and verification of each step.
* Understanding C++ Specifics: Features like std::vector operations and C++ references demanded a deep dive into C++ specific constructs.