CS 300 Module Two Assignment Guidelines and Rubric

Overview

For this assignment, you'll use information from a municipal government data feed that contains bids submitted for property auctions. All materials for this assignment can be found in the Supporting Materials section below. The data set is provided in two CSV files:

- eBid_Monthly_Sales.csv (larger set of 12,023 bids)
- eBid_Monthly_Sales_Dec_2016.csv (smaller set of 76 bids)

You will explore sorting algorithms by implementing a selection sort and a quicksort of a vector of bids from a CSV file.

You will be given a starter console program that uses a menu to enable testing of the sort logic you will complete. The console program also allows you to pass in the path to the bids CSV file to be loaded, enabling you to try both files. In this version, the following menu is presented when the program is run:

Menu:

1. Load Bids
2. Display All Bids
3. Selection Sort All Bids
4. QuickSort All Bids

9. Exit

Enter choice:

The VectorSorting.cpp program is partially completed. The program contains empty methods representing the programming interface used to interact with the vector. You must add logic to the methods to implement the necessary behavior. Here are the methods in VectorSorting.cpp that you must complete:

```
void selectionSort(vector& bids)


void quickSort(vector& bids, int begin, int end)
int partition(vector& bids, int begin, int end)
```

Directions

You must perform the following steps to complete this activity:

Setup: Begin by creating a new C++ project with the project type ”Hello World C++ Project“. For help setting up your project in Visual Studio C++, refer to the Apporto Visual Studio Setup Instructions and Tips in the Module One Resources section. Name the project “VectorSorting”.

Task 1: Implement the selection sort algorithm:

a. Code the selection sort logic using “bid.title” as the sort field.

b. Invoke the selectionSort() method from the main() method including collecting and reporting timing results.

Task 2: Implement the quicksort algorithm:

a. Code the quicksort logic using “bid.title” as the sort field.

b. Invoke the quickSort() method from the main() method including collecting and reporting timing results.

Here is sample output from running the completed program:

```
> ./VectorSorting ~/Downloads/eBid_Monthly_Sales.csv
> VectorSorting.exe Downloads\eBid_Monthly_Sales.csv
```

Loading bids from CSV and performing a selection sort:

Example Input	Choice: 1	Choice: 3
Display	Menu: 1. Load Bids 2. Display All Bids 3. Selection Sort All Bids 4. QuickSort All Bids 9. Exit Enter choice: 1	Menu: 1. Load Bids 2. Display All Bids 3. Selection Sort All Bids 4. QuickSort All Bids 9. Exit Enter choice: 3
Output	Loading CSV file eBid_Monthly_Sales.csv 12023 bids read time: 12023 clock ticks time: 0.173945 seconds	12023 bids sorted time: 10623604 clock ticks time: 10.6236 seconds

Loading bids from CSV and performing a quicksort:

Example Input	Choice: 1	Choice: 4
Display	Menu: 1. Load Bids 2. Display All Bids 3. Selection Sort All Bids 4. QuickSort All Bids 9. Exit Enter choice: 1	Menu: 1. Load Bids 2. Display All Bids 3. Selection Sort All Bids 4. QuickSort All Bids 9. Exit Enter choice: 4
Output	Loading CSV file eBid_Monthly_Sales.csv 12023 bids read time: 174985 clock ticks time: 0.174985 seconds	12023 bids sorted time: 47964 clock ticks time: 0.047964 seconds

Notice the huge time difference in sorting 12,000 records: 10.6 seconds compared to just 0.04 seconds!

Specifically, you must address the following rubric criteria:

- **Code Reflection:** Describe the purpose of code, techniques implemented to solve problems, challenges encountered, and approaches to overcome the challenges.
- **Pseudocode or Flowchart:** Provide a pseudocode or flowchart description of the code that is clear and understandable and captures accurate logic to translate to the programming language.
- **Specifications and Correctness:** Source code must meet its specifications and behave as desired. Correct code produces the correct output as defined by the data and problem. However, you should also produce fully functioning code with no errors that aligns with as many of the specifications as possible. You should write your code in a way that the submitted file executes, even if it does not produce the correct output. You will be given credit for partially correct output that can be viewed and seen to be partially correct.
- **Annotation and Documentation:** All code should also be well commented. Commenting is a practiced art that requires striking a balance between commenting everything, which adds unneeded noise to the code, and commenting nothing. Well-annotated code requires you to perform the following actions:
    - Explain the purpose of lines or sections of your code, detailing the approach and method you took to achieve a specific task in the code.
    - Document any section of code that is producing errors or incorrect results.
- **Modular and Reusable:** Programmers should develop code that is modular and reusable. Code is more flexible and maintainable if it contains functionality and responsibility in distinct methods. Your code should adhere to the single responsibility principle. Classes and methods should do only one job. If you can use a different method without changing other parts of your code, you have succeeded in creating modular methods.

- **Readability:** Code needs to be readable to a knowledgeable programmer. In this course, readable code requires the following characteristics:
  - Consistent, appropriate whitespace (blank lines, spaces) and indentation to separate distinct parts of the code and operations
  - Explicit, consistent variable names, which should clearly indicate the data they hold and be formatted consistently, for example, numOrders (camelCase) or item_cost (underscored)
  - Organized structure and clear design that separates components with different responsibilities or grouping-related code into blocks

What to Submit

To complete this assignment, submit the VectorSorting.cpp **code files** and a **code reflection and associated pseudocode or flowchart** written portion. Your written portion should be 1–2 paragraphs in length.

Supporting Materials

The following resource may help support your work on the project:

Resource: Vector Sorting Assignment Student Files

Download this zipped file folder to begin your assignment. The data sets you will use in this assignment are provided in these CSV files:

- eBid_Monthly_Sales.csv (larger set of 12,023 bids)
- eBid_Monthly_Sales_Dec_2016.csv (smaller set of 77 bids)
- VectorSorting.cpp program, which is a partially completed program that you can use as a starting point for the assignment

Module Two Assignment Rubric

Criteria	Proficient (100%)	Needs Improvement (70%)	Not Evident (0%)	Value
Code Reflection	Describes purpose of code, techniques implemented to solve problem, challenges encountered, and approaches to overcome the challenges	Lacks details in code purpose, techniques implemented, or challenges encountered	Does not explain purpose of code, techniques used, or challenges encountered	25
Pseudocode or Flowchart	Pseudocode or flowchart is clear and understandable and captures accurate logic to translate to the programming language	Pseudocode or flowchart has errors or omissions that affect its clarity or understandability, or the logic to translate to the programming language is inaccurate or incomplete	Pseudocode or flowchart does not contain the logic to translate to the programming language	10
Specifications and Correctness: Algorithm	All algorithm specifications are met completely and function in all cases	Details of the specifications are violated, or program often exhibits incorrect behavior	Program only functions correctly in very limited cases or not at all	20

Criteria	Proficient (100%)	Needs Improvement (70%)	Not Evident (0%)	Value
Specifications and Correctness: Data Structure	All data structure specifications are met completely and function in all cases	Details of the specifications are violated, or program often exhibits incorrect behavior	Program only functions correctly in very limited cases or not at all	20
Annotation and Documentation	Code annotations explain and facilitate navigation of the code	Comments provide little assistance with understanding the code	Code annotations do not explain the code or do not facilitate navigation of code, or code is not fully or logically annotated	10
Modular and Reusable	Methods are limited in scope and responsibility, and both algorithms and data structures are implemented in such a way that they can be reused in other programs	Methods have errors in scope or responsibility, or algorithms or data structure are overly tied to the specific program	No attempt was made to develop modular or reusable code	10
Readability	Code follows proper syntax and demonstrates deliberate attention spacing, whitespace, and variable naming	Code contains variations from established syntax and conventions	Code contains significant variations from established syntax and conventions	5
			Total:	100%