Lab 4-2 Vector Sorting Assignment Details

The focus of these problems will be working with information extracted from a municipal government data feed containing bids submitted for auction of property. The data set is provided in two comma-separated files:

1. eBid\_Monthly\_Sales.csv (larger set of 17,937 bids)
2. eBid\_Monthly\_Sales\_Dec\_2016.csv (smaller set of 179 bids)

This assignment is designed to explore sorting algorithms by implementing both a selection sort and quicksort of a vector of bids loaded from a CSV file.

We provide a starter console program that uses a menu to enable testing of the sort logic you will complete. It 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 - it contains empty methods representing the programming interface used to interact with the linked list. You will need to add logic to the methods to implement the necessary behavior. Here are the methods in VectorSorting.cpp that you have to complete:

void selectionSort(vector<Bid>& bids)

void quickSort(vector<Bid>& bids, int begin, int end)

int partition(vector<Bid>& bids, int begin, int end)

You will need to perform the following steps to complete this activity:

Setup: Begin by creating a new C++ Project with a Project Type of "Hello World C++ Project"

1. Name the project ‘VectorSorting’, remember to pick the correct compiler in **Toolchains** and click Finish. This will create a simple VectorSorting.cpp source file under the **/src** directory.
2. Download the starter program files and copy them to the project’s **/src** directory, replacing the existing auto-generated one. Remember to right-click on the project in the Project Explorer pane on the left and **'Refresh'** the project so it adds all the new files to the src folder underneath.
3. Because this activity uses C++ 11 features you must follow the instructions under “C++ Compiler Version” in the *C++ Development Installation guide* to add **-std=c++11** compiler switch to the Miscellaneous settings.

Task 1: Implement the selection sort algorithm:

1. Code the selection sort logic using bid.title as the sort field.
2. Invoke the selectionSort() method from the main() method including collecting and reporting timing results.

Task 2: Implement the quicksort algorithm:

1. Code the quicksort logic using bid.title as the sort field.
2. 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

**Load 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  17937 bids read  time: 173945 clock ticks  time: 0.173945 seconds | 17937 bids sorted  time: 10623604 clock ticks  time: 10.6236 seconds |

**Load 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: 4 | 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  17937 bids read  time: 174985 clock ticks  time: 0.174985 seconds | 17937 bids sorted  time: 47964 clock ticks  time: 0.047964 seconds |

Note the dramatic difference in the time it takes to sort nearly 18,000 records - 10.6 seconds versus 4 *hundredths* of a second!