CS 300 Module Four Assignment Guidelines and Rubric

Overview

For this assignment, you will use information from a municipal government data feed that contains bids submitted for property auctions. All materials for this lab 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)

In this assignment, you will explore hash tables. You will implement a hash table to hold a collection of bids loaded from a CSV file. You will be given a starter console program that uses a menu to enable testing of the hash table 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. Find Bid
4. Remove Bid

9. Exit

Enter choice:

The HashTable.cpp program is partially completed. The program contains empty methods representing the programming interface used to interact with a hash table. You must add logic to the methods to implement the necessary behavior. Here is the public API for HashTable.cpp that you must complete:

```
public:
```

1. `HashTable();`
2. `virtual ~HashTable();`
3. `Hash;`
4. `void Insert(Bid bid);`
5. `void PrintAll();`
6. `void Remove(string bidId);`

7. `Bid Search(string bidId);`

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 “HashTable”.

Task 1: Task 1 has already been done for you. Hint: You may be able to reuse portions of your code from previous assignments to save you time. Look for places where you have implemented similar functions to perform these tasks and take advantage of existing code’s reusability whenever possible. You may have to modify the code slightly for each assignment, especially when working with vectors for storage or a node structure for a linked list. Reusing code from these labs may save you time.

Task 2: Initialize the structures used to hold bids.

Task 3: Implement code to free storage when a class is destroyed.

Task 4: Implement code to calculate a hash value using the bid ID as the source for calculating the key.

Task 5: Implement code to insert a bid. Be sure to check for key collisions and use the chaining technique with a linked list to store the additional bids.

Task 6: Implement code to print all bids.

Task 7: Implement code to remove a bid.

Task 8: Implement code to search for and return a bid.

Here is sample output from running the completed program:

```
> ./HashTable ~/Downloads/eBid_Monthly_Sales_Dec_2016.csv
> HashTable.exe Downloads\eBid_Monthly_Sales_Dec_2016.csv
```

Load bids from CSV and display the hash table contents:

Example Input	Choice: 1	Choice: 2

Display	Menu: 1. Load Bids 2. Display All Bids 3. Find Bid 4. Remove Bid 9. Exit Enter choice: 1	Menu: 1. Load Bids 2. Display All Bids 3. Find Bid 4. Remove Bid 9. Exit Enter choice: 2
Output	Loading CSV file eBid_Monthly_Sales.csv 12023 bids read time: 3069 clock ticks time: 0.3069 seconds	Key 2: 98094 | Credenza | 57 | General Fund 2: 98273> | Nike Tennis Shoes Size: 11.5 | 84 | Enterprise Key 5: 98276 | Nike Tennis Shoes Size: 11.5 | 83.99 | Enterprise Key 8: 98279 | Nike Tennis Shoes Size: 11 | 51.57 | Enterprise Key 10: 98102 | Battery Cart | 42 | Enterprise Key 12: 98104 | 3 Ticket Booths | 395.01 | Enterprise 12: 98283 | Jordan Tennis Shoes Size: 11 | 160 | Enterprise Key 13: 98105 | 2 PS4 Games | 11 | Enterprise 13: 98284 | Jordan Tennis Shoes Size: 11 | 89.01 | Enterprise ... ... Key 176: 98268 | Dayton Pallet Jack | 78.85 | Enterprise Key 177: 98269 | 5 Extron Control Systems | 25 | General Fund

Note that Keys 2, 12, and 13 indicate that key collisions occurred.

Finding and removing an existing bid:

Example Input	Choice: 3	Choice: 4
Display	Menu: 1. Load Bids 2. Display All Bids 3. Find Bid 4. Remove Bid 9. Exit Enter choice: 3	Menu: 1. Load Bids 2. Display All Bids 3. Find Bid 4. Remove Bid 9. Exit Enter choice: 4
Output	98223: Chair| 71.88 General Fund time: 59 clock ticks time: 5.9e-05 seconds	{no output shown}

Finding a bid that no longer exists:

Example Input	Choice: 3	Choice: 9
Display	Menu: 1. Load Bids 2. Display All Bids 3. Find Bid 4. Remove Bid 9. Exit Enter choice: 3	Menu: 1. Load Bids 2. Display All Bids 3. Find Bid 4. Remove Bid 9. Exit Enter choice: 9

Output	Bid Id 98129 not found. time: 12 clock ticks time: 1.2e-05 seconds	Good bye.

Specifically, you must address the following rubric criteria:

- **Code Reflection:** Briefly 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 lab assignment, submit the HashTable.cpp **code files** and a **code reflection and associated pseudocode or flowchart**. Your written portion should be 1–2 paragraphs in length.

Supporting Materials

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

Resource: Hash Table Assignment Student Files

Download this zipped file folder to begin your lab 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 76 bids)
- HashTable.cpp program, which is a partially completed program that you can use as a starting point for the assignment

Module Four 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
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

Criteria	Proficient (100%)	Needs Improvement (70%)	Not Evident (0%)	Value
			Total:	100%