**Danny Forte**

**CS-300 Analysis and Design**

**March 29, 2025**

**HashTable: Code Reflection**

This assignment continued to build upon our previous works and the note in the assignment for us to look for pieces of code that we can reuse with adjustments if needed was helpful. We once again load a CSV file and ask for it to display all bids, we asked for it to find a bid and then to be able to delete a bid. The main difference for this assignment was the use of the hash tables for storing and locating the data. The hash table stores the value/key. For us, the bids information of the value and id is the key that is being mapped to our table. Using a hash table you can perform many functions such as find , add or delete an index.

The key is hashed to an index location using the modulo operator "%" and the table size. This calculation creates a remainder and that is used as the index location to store the information. The bidId is also a unique value and that will create a unique index for each item. We did learn about collisions and how we can handle them. A collision is when more than one item shares hash key. Chaining is a method that will help us to reduce collisions when items are assigned to the same location.

I am also grateful for our continued practice with VS Code. I am gaining more confidence with my understanding of this IDE. I had no problems with my ability to create the project, add the existing files to it, and then finish the code, compile, build and even debug it.

**PsuedoCode**

While choice IS NOT equal to 9

Main

Display user menu:

1. Load Bids

2. Display All Bids

3. Find Bid

4. Remove Bid

5. Insert Bid

9. Exit

Ask for User input and store choice for menu selection

If Choice = 1. Load Bids

Start Clock and store ticks

Call loadBids to open and store CSV data

Output number of records found in CSV file

Stop the clock

Display time needed to read CSV file

Break

If Choice = 2. Display All Bids

Loop through bids vector

Display Bids

Break

If Choice = 3. Find Bid

Start Clock and store ticks

Set bid equal to search() method and pass in bidkey as the paramenter

Call search()

Search() will complete:

Set key equal to hashed bidId

Set new pointer called node equal to ref of nodes element at its keyed position

If node is not equal to null pointer And node key Is Not equal to UINT\_MAX

And node bidId compared to passed bidId is a match

Return node bidId;

If node Is equal to null pointer Or node key is equal to UINT\_MAX

Return bid;

While node Is Not equal to null pointer

If node key Is not equal to UINT\_MAX And node bidId compare bidId is a match

Return node bidId;

Set node equal to next node;

Return bid;

Stop the clock

If bidId Is not empty

Call displayBid() method and pass in bid;

Else

Display bid not found message

Display time

Break

If choice = 4. Remove bid

Call Remove() method with passed bidId;

Remove() method will complete

Set key equal to hashed bidId

Erase node starting at the beginning with the hashed key

Resize nodes table

If choice = 5. Insert Bid

Create getBid()method

getBid()method collects input from user

Set bid equal to getBid() method

Call Insert() method and pass in bid

Insert method will complete

Set key to hashed bidId

If nodes keys at the its position IS equal to UINT\_MAX

Create new node

Set new node equal to Node with passed in bid and key

Set nodes at key position equal to new node

Else

Create new pointer currNode

Set currNode equal to reference nodes at key position

Create new pointer newNode

Set newNode equal to a new Node with passed in bid and key

While next currNode Is Not equal to NULL

Point currNode to the next currNode

Set currNode equal to the newNode

Set currNode next equal to the newNode

Resize nodes table

Call dispalyBid() and pass in bid

Break

Else

Display Goodbye

Return 0