Philip Trinh

SNHU/CS-300

3/24/2023

**Reflection and Pseudocode module 4 Hash Table**

In this assignment, I am working with information from a municipal government data feed containing bids submitted for auction of property. I was provided an incomplete HashTable.cpp program in which I need to make changes to Hash Table class, insert, print, remove, search methods. To tackle this project, I reviewed the ZYBook lessons on chapter 5 which contain Hash Table, insert, print, remove, search methods examples. The hash table important is to set up key and modular it with the size of the item expected to contain to make sure that all can fit into slot prepared. Insert method will add a new bid to the list by finding the slot that is empty or not in use and insert it into that slot. printAll will print all the bids that are currently in the list, search will search for a specified bid, remove will remove the specified bid. I enjoyed following the instructor provided in the video lecture and read the comment instruction provided from HashTable.ccp so I did not run into any issue besides a bit confused result. I was able to use g++ to compile the code with no problem, load and display bids work fine. Option 3 and 4 for find and remove are different than expected. Option 3 return bid 98109 not found and option 4 to remove bid did not seem to do anything. However, further reading about the expected return and go through the code myself, I didn’t see any issue and maybe the problem lies in not having an insert option on the menu which isn’t require to be fix for this assignment so I left it alone.

**Pseudocode**

**Create Hash table, Insert, print All, Remove, Search**

Declare clash HashTable and define structure to hold bids, initialize bid and key

Constructor

Initalize node structure by resizing tableSize

Destructor

Erase nodes from beginning to free storage when class is destroyed

Implement logic to calculate hash value based on return of key modular with tableSize

Insert method

Create the key for the given bid, take bid, have bid ID

Retrieve node using key

If no entry found for node

Assign current node to the key position

Else if node is found

If previous node is empty

Assign previous node with key, bid, and set next one to empty

Else if not found and have a collision

Find the next open slot

Add new node to the end

PrintAll method

Use for loop to iterate node form begin to end

If key is not occupied

Print out key, bid title, bid fund, then move to next one

If node is not empty

Print out key, bid ID, bid title, bid amount, bid fund, then move to next one

Remove method

Using if to erase node begin and key, find collision and remove it only

Compare if bid id from current node and bidId searching for matching then remove

If next node is empty, set it to maximum value allow and return so next time it can replace this key with a value

Else skip the current node and point to next node, return

Else if first node not match what is searching for then current node will point to next node, previous node is equal to the node

While the current node is not empty, compare bid ID searching with current one, if matching then skip this node, set previous node equal to next, and current is next,

Delete this current node to free storage

Set current node to empty and return it

Search method

create the key for the given bid

use key to find node

if entry found for the key, key not equal to UINT\_MAX, slot is not empty, and first item return the bidId searching for

return that node bid

if no entry found for the key

return bid

while node not empty

if the current node matches

return it

node is equal to next node

return the bid