Hash Tables introduced a new way of doing things that I wasn’t immediately ready for, for most of the assignment it made sense until I got to the print all function as I couldn’t figure out how to only output the nodes that had anything in them. The other part was my original implementation wasn’t a smart solution where all I was doing was iterating my for loop to increase by one as it approached the tables size. This was a new challenge I had not anticipated that sent me to all corners of the internet trying to find my solution to the issue which in almost all cases they had the same solution as me to use slower and less efficient for loop that had no way of knowing where the nodes began or ended, but with some good help for the instructor it all started to make sense and I was able to make the table display properly.

**Pseudocode for this project   
Initializing the Hash Table**  
-tablesize is set to the default size which in this case is set to 179  
-We then initialize the nodes to be the same size as the tablesize.

**Resizing the Nodes  
-**We first use the tablesize which was defined earlier we then have it equal to size  
-Then we resize the nodes  
  
**Erasing the nodes**-we erase the nodes starting from the beginning  
  
**Inserting A Bid**-First we create a key using hash(atoi(bid.bidId.c\_str()))  
-then we get the node using the ke  
-if there is not entry found for the key we make a new node for that keys position  
-else   
-If the node is not used assign the old node key to UNIT\_MAX then we set to key, set old node to bid and old node next to null pointer  
-else  
-while loop looks for an unoccupied node to give it the bid and the key  
-after the while loop adds a new node to the end

**Printing all of the bids loaded**-for loop that starts at the beginning of the nodes and keeps iterating until it reaches the end of the nodes  
-if the key is not equal to UINT\_MAX output the key, bidId, title, amount and fund  
-make the node equal to the next iteration  
-while node is not equal to null pointer  
- output the key, bidId, title, amount and fund  
-set the node to equal the next node

HashTable searching  
-if the node is not equal to null pointer and node-> key is equal to UINT\_MAX return the node->bid  
-if the node is equal to null pointer or node-> key is equal to UINT\_MAX return the bid  
-while node is not equal to null pointer  
-if node-> key is not equal to UINT\_MAX and node->bid.bidId.compare(bidId) == 0  
-return node -> bide  
-make the node equal to the next node  
-return bid