Daniel Loranger

CS-300

Week 3 assignment

* Code Reflection – Overall I found the code fairly straight forward to manipulate and generate, with minor struggles to migrate the code and methods from the zybooks learnings into functional code.

The only thing I found slightly challenging was performing the removal search as it was difficult to keep in mind that I was searching for a future item, not the current item. This was required to ensure I could update the current nodes next pointer. If I instead searched against the current node, I had no easy way to go back one step and fix the previous pointer.

I did see that a displayBid() function was available for printing the results, so rather than manually creating the outputs separately, I reused this function. If this code was a real program rather than an exercise, I would have made this a private function of the class instead of being a standalone test function.

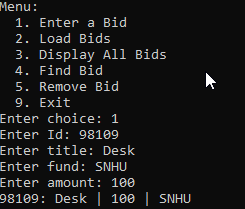
I also updated the class functions that should not edit data (display and search) to be declared const to ensure they don’t change any data in future edits by accident.

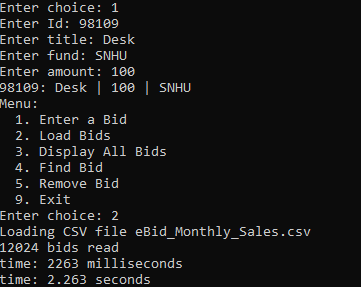
* Pseudo-code

For brevity, existing static functions for testing are not detailed, only the functions that are edited are covered with below pseudo code. Exception is the main function which is shown for flow understanding inclusively.

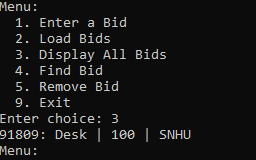
* + Declare variables
  + Define class including constructor initializations (fixme 1)
  + Append function (fixme 2)
    - Special case for empty list
    - Update the tail pointer for non-empty list
    - Update the tail pointer
    - Increment size
  + Prepend function (fixme 3)
    - Special case for empty list
      * Insert using append function
    - Set new node.next to point to head
    - Update head pointer
    - Increment size
  + PrintList function (fixme 4)
    - Update declaration to be a const
    - Special case for empty list
    - While loop through all list items,
      * print using displayBid()
  + Remove function (fixme 5)
    - Special case for empty list
    - Special case for empty bidId
    - Special case for head
      * Special case for head == tail
        + Empty head and tail pointers
      * Update head pointer to next
      * Decrement size
    - While (not logic) loop through all list items, looking ahead 1 position for match
      * Special case for tail position
      * Update current node to skip next position
    - Match found
      * Special case for tail position
        + Set next pointer to null
        + Update tail pointer
      * Update next to next.next to skip the matched item
  + Search function (fixme 6)
    - Update declaration to make function a const
    - While loop thru all list items,
      * compare current item & return bid if matched
    - no match found, return empty bid structure
  + Size function
    - Update declaration to make function a const
* Static functions (unchanged)
  + displayBid()
  + getBid()
  + loadBids()
  + strToDouble
  + Main ()
    - Handle command line arguments
    - declare an empty time place holder
    - declare an empty linked list
    - declare an empty bid
    - while loop
      * display user menu
      * handle user selection (case struct)

Specifications and Correctness

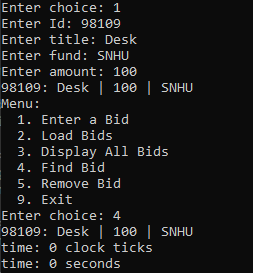
* Add a bid (menu choice 1) – PASS, but note that using an entry with more than 1 word for fund will cause the function to fail. This was provided code so was not corrected…  
  
* Load Bids (menu choice 2) – PASS, note count is +1 since step 1 added a new bid



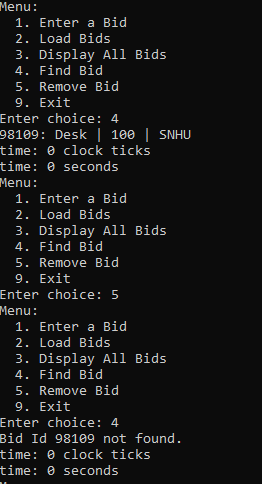
* Display all bids (menu choice 3) – Pass



* Find Bid (menu choice 4) – PASS



* Remove Bid (choice 5) – Pass



* Annotation and Documentation – Code is commented as necessary
* Modular and Reusable – code is improved, reused existing code to eliminate redundancy
* Readability
  + Appropriate white space and formatting is used
  + camelCase naming convention is used (aligned to original format)
  + Structure – follows provided straw-man outline