The code for the module three “LinkedLists” assignment is reflected upon in the following.

This C++ program is assembled with a collection of bids using a linked list data structure. We use the Bid structure to hold all the bid information. The bid information includes bidId, title, fund, & amount. The LinkedList class manages a dynamic list of the nodes, where each node holds a Bid object. The program provides core operations like appending bids to the end of the list, prepending them to the start, searching for a specific bid by bidId, removing bids, and displaying all bids in the list.

While it is a functional program, the reliance on using a linkedlist makes some of it’s operations like searching, less efficient.

Define Bid Structure

* + Contains fields: bidId, title, fund, amount
  + Constructor initializes amount to 0.0

Define Node Structure (used within LinkedList)

* + Contains: bid (of type Bid), next (pointer to the next Node)
  + Constructor initializes next to nullptr

Define LinkedList Class

* + Variables:
    - head (points to first Node)
    - tail (points to last Node)
    - size (keeps track of the number of Nodes)
  + Constructor: Initializes head and tail to nullptr, size to 0
  + Destructor: Deletes all nodes in the list

LinkedList Methods:

* + Append(bid):
    - Create a new Node with bid
    - If head is nullptr, set both head and tail to the new Node
    - Else, set tail->next to the new Node and update tail
    - Increment size
  + Prepend(bid):
    - Create a new Node with bid
    - If head is not nullptr, set newNode->next to head, then update head
    - If head is nullptr, append the bid
    - Increment size
  + PrintList():
    - Start at head and traverse the list
    - For each Node, print bidId, title, amount, fund
  + Remove(bidId):
    - Traverse the list, searching for the Node where bidId matches
    - Special case: if the Node is the head, update head to head->next
    - If found, update pointers to remove the Node and free memory
    - Decrease size
  + Search(bidId):
    - Start at head, traverse the list, and check each Node’s bidId
    - If a match is found, return the Bid
    - If not found, return an empty Bid
  + Size():
    - Return the current number of Nodes (size)

Helper Functions:

* + Print the details of a Bid
  + Prompt the user to input bidId, title, fund, and amount, then return a Bid
  + loadBids(csvPath, list):
    - Open the CSV file and read each row
    - Create a Bid from the row data
    - Append the Bid to the list

Main Function:

* + Parse command line arguments to set csvPath and bidKey
  + Create a LinkedList object bidList
  + Display a menu of options to the user:
    - Option 1: Get a Bid from the user and append it to the list
    - Option 2: Load bids from a CSV file into the list
    - Option 3: Print all bids in the list
    - Option 4: Search for a Bid using bidKey and display it
    - Option 5: Remove a Bid using bidKey
    - Option 9: Exit the program