# Linked List Assignment – Code Reflection and Pseudocode

## Code Reflection

The goal of this project was to build a singly linked list to manage auction bids loaded from a CSV file and test the list through a console menu. The LinkedList class uses an internal Node structure containing a Bid and a pointer to the next node, with head and tail pointers plus a counter for list size. Each public method matches the assignment specification: Append links to the tail in O(1), Prepend inserts at the head, PrintList traverses and outputs bids, Search scans for a matching ID, and Remove carefully handles head, middle, and tail cases. The destructor walks the list to delete all nodes and avoid memory leaks. For loading data, a CSV parser was added to read headers and map the dataset fields (ArticleID, ArticleTitle, Fund, and WinningBid) to the Bid structure, while parsing currency strings into doubles.

Challenges included handling CSV fields with commas inside quotes and ensuring the removal function updated head and tail correctly when deleting special cases such as the first or last node. The CSV parsing was solved by writing a character-by-character parser that toggles a quote flag to differentiate between commas inside and outside of quotes. The removal edge cases were solved by explicitly checking whether the node to delete was the head or tail and updating pointers accordingly. The result is a modular, reusable, and correct implementation that meets the requirements.

## Pseudocode

STRUCT Bid:  
 bidId: string  
 title: string  
 fund: string  
 amount: double  
  
CLASS LinkedList:  
 STRUCT Node:  
 bid: Bid  
 next: Node\*  
  
 MEMBERS:  
 head: Node\* = null  
 tail: Node\* = null  
 count: int = 0  
  
 METHOD Append(bid):  
 node = new Node(bid)  
 IF head == null:  
 head = tail = node  
 ELSE:  
 tail.next = node  
 tail = node  
 count++  
  
 METHOD Prepend(bid):  
 node = new Node(bid)  
 IF head == null:  
 head = tail = node  
 ELSE:  
 node.next = head  
 head = node  
 count++  
  
 METHOD PrintList():  
 cur = head  
 WHILE cur != null:  
 display(cur.bid)  
 cur = cur.next  
  
 METHOD Remove(bidId):  
 IF head == null: RETURN  
 IF head.bidId == bidId:  
 tmp = head  
 head = head.next  
 IF tmp == tail: tail = head  
 delete tmp  
 count--  
 RETURN  
 prev = head  
 cur = head.next  
 WHILE cur != null:  
 IF cur.bidId == bidId:  
 prev.next = cur.next  
 IF cur == tail: tail = prev  
 delete cur  
 count--  
 RETURN  
 prev = cur  
 cur = cur.next  
  
 METHOD Search(bidId) -> Bid:  
 cur = head  
 WHILE cur != null:  
 IF cur.bidId == bidId: RETURN cur.bid  
 cur = cur.next  
 RETURN empty Bid  
  
MAIN:  
 list = new LinkedList  
 REPEAT until choice == 9:  
 print menu  
 read choice  
 CASE 1: read Bid from user; Append; display  
 CASE 2: load CSV; Append each row as Bid  
 CASE 3: PrintList  
 CASE 4: prompt id; Search; display or not found  
 CASE 5: prompt id; Remove  
 CASE 9: exit