**Linked Lists Code Reflection**

**By: Darrell Walker**

**Purpose of the Code**

The code is designed to create a simple linked list in C++. A linked list is a way to store a collection of data elements (in this case, bids) where each element points to the next one in the sequence. This linked list holds information about bids, including the bid ID, title, fund, and amount. The code allows you to add bids to the end or beginning of the list, remove a bid by its ID, search for a bid by its ID, and print out all the bids in the list.

**Techniques Implemented to Solve Problems**

The code uses several techniques to manage the linked list. First, it defines a Node structure to hold each bid and a pointer to the next node. The LinkedList class manages these nodes. It keeps track of the head (the first node) and the tail (the last node) of the list, as well as the size of the list. The Append method adds a new bid to the end of the list, while the Prepend method adds a new bid to the beginning. The Remove method removes a bid with a specific ID, and the Search method finds a bid by its ID. The Print List method prints all the bids in the list.

**Challenges Encountered**

One challenge was managing memory correctly. Since the nodes are dynamically allocated, it's important to ensure they are properly deleted to avoid memory leaks. Another challenge was handling edge cases, such as when the list is empty or when the bid to be removed is at the head of the list. Manipulating pointers correctly when adding or removing nodes was also tricky and required careful attention to detail.

**Approaches to Overcome the Challenges**

To manage memory, the destructor was implemented to delete all nodes when the list is destroyed. Conditional checks were added to handle edge cases in methods like Remove and Search. The code was regularly tested with different scenarios to identify and fix bugs related to pointer manipulation and list management.

**Task 1: Create an Internal Structure for List Entries and Housekeeping Variables**

**Purpose**: Define a structure to hold bid entries and any other variables needed to manage the list.

Class LinkedList

Internal Structure Node

Bid bid

Node\* next

Constructor Node()

Set next to nullptr

Constructor Node(Bid aBid)

Set bid to aBid

Set next to nullptr

End Structure

Node\* head

Node\* tail

int size

End Class

**Task 2: Initialize Housekeeping Variables**

**Purpose**: Initialize the housekeeping variables in the constructor.

Constructor LinkedList()

Set head to nullptr

Set tail to nullptr

Set size to 0

End Constructor

**Task 3: Implement Append Logic**

**Purpose**: Add a new bid to the end of the linked list.

Function Append(bid)

Create a new Node with bid

If head is nullptr Then

Set head to new Node

Set tail to new Node

Else

Set tail->next to new Node

Set tail to new Node

End If

Increment size by 1

End Function

**Task 4: Implement Prepend Logic**

**Purpose**: Add a new bid to the start of the linked list.

Function Prepend(bid)

Create a new Node with bid

If head is not nullptr Then

Set new Node->next to head

End If

Set head to new Node

If tail is nullptr Then

Set tail to new Node

End If

Increment size by 1

End Function

**Task 5: Implement Print Logic**

**Purpose**: Print all bids in the linked list.

Function PrintList()

Set current to head

While current is not nullptr

Print current->bid.bidId, current->bid.title, current->bid.amount, current->bid.fund

Set current to current->next

End While

End Function

**Task 6: Implement Remove Logic**

**Purpose**: Remove a bid with a specific bid ID from the linked list.

Function Remove(bidId)

If head is not nullptr AND head->bid.bidId equals bidId Then

Set temp to head

Set head to head->next

Delete temp

Decrement size by 1

Return

End If

Set current to head

While current is not nullptr AND current->next is not nullptr

If current->next->bid.bidId equals bidId Then

Set temp to current->next

Set current->next to temp->next

Delete temp

Decrement size by 1

Return

End If

Set current to current->next

End While

End Function

**Task 7: Implement Search Logic**

**Purpose**: Search for a bid with a specific bid ID in the linked list and return it.

Function Search(bidId) -> Bid

If head is not nullptr AND head->bid.bidId equals bidId Then

Return head->bid

End If

Set current to head

While current is not nullptr

If current->bid.bidId equals bidId Then

Return current->bid

End If

Set current to current->next

End While

Return an empty Bid

End Function