**Seema Chavesta**

**Sep 22, 2024**

**Code Reflection**

This implementation defines a basic linked list with operations such as appending, prepending, searching, removing, and printing elements. The linked list is made up of Node structures, each containing a Bid object and a pointer to the next node in the list. The list uses two key pointers, head and tail, to maintain the beginning and end of the list.

1. **Data Structure**:
   * The Node struct contains:
     + A Bid object, which has fields like bidId, title, fund, and amount.
     + A pointer to the next Node (next).
2. **Key Operations**:
   * **Append**: Adds a new Node containing the Bid at the end of the list.
   * **Prepend**: Adds a new Node at the beginning of the list.
   * **Search**: Traverses the list to find a Node with the given bidId.
   * **Remove**: Finds and deletes a Node with the given bidId.
   * **Print**: Outputs the entire list, displaying the details of each Bid.
3. **Linked List Properties**:
   * A head pointer is maintained for the start of the list.
   * A tail pointer allows for efficient appending at the end.
   * A size counter tracks the number of elements.
4. **Efficiency**:
   * The Append operation is efficient as it uses the tail pointer, so no traversal is needed.
   * The Search and Remove operations traverse the list, making them O(n) operations.
   * Prepend is an O(1) operation since it only modifies the head pointer.

**Pseudocode for Linked List Operations**

**Append a Bid to the list**

Procedure Append(bid)

Create a new Node(newNode) with bid

If head is NULL then

Set head to newNode

Set tail to newNode

Else

Set tail.next to newNode

Set tail to newNode

End If

Increment size

End Procedure

**2. Prepend a Bid to the list**

Procedure Prepend(bid)

Create a new Node(newNode) with bid

If head is NULL then

Set head to newNode

Set tail to newNode

Else

Set newNode.next to head

Set head to newNode

End If

Increment size

End Procedure

**Search for a Bid by bidId**

Procedure Search(bidId)

Set current to head

While current is not NULL

If current.bid.bidId equals bidId then

Return current.bid

End If

Set current to current.next

End While

Return an empty Bid (not found)

End Procedure

**Remove a Bid by bidId**

Procedure Remove(bidId)

If head is NULL then

Return (list is empty)

End If

If head.bid.bidId equals bidId then

Set temp to head

Set head to head.next

Delete temp

Decrement size

Return

End If

Set current to head

While current.next is not NULL

If current.next.bid.bidId equals bidId then

Set temp to current.next

Set current.next to current.next.next

Delete temp

Decrement size

Return

End If

Set current to current.next

End While

End Procedure

**Print all bids in the list**

Procedure PrintList()

Set current to head

While current is not NULL

Display current.bid

Set current to current.next

End While

End Procedure

**Get the size of the list**

Procedure Size()

Return size

End Procedure