**Module 3 Reflection and Pseudocode**

Jermaine Wiggins

Southern New Hampshire University

CS300 Analysis and Design

Prof. Sathish Gopalakrishnan

Jan 26, 2025

**Module 3 Reflection and Pseudocode**

Like last week before starting the ZyBooks assignment, I glanced at the instructions, and it seemed very challenging, specifically because I’m not the best when it comes to pointers. This module did give me a better understanding of pointers through traversing, removing, appending, and prepending linked list.

The purpose of this program is to load a collection of bids into a linked list, from there the user can append, prepend, search, and print all bids. The first fix-me was to initialize housekeeping variables, the head and tail nodes were both set to null pointer, meaning the linked list is empty.

The next fix-me was to implement the append method, which takes bid as parameter and adds a bid to the linked list. First we create a new node pointer and pass bid as a parameter which calls the constructor Node(bid aBid). Next an if statement check if the head is null, if it is, head and tail is set equal to new node, meaning the list is empty. If the head isn’t null, then the tail next is set to new node and tail is updated to new node. Finally, size is increased by 1.

**Append Pseudocode**

**CREATE** new node

**IF** head is null

**SET** head and tail to new node

**ELSE**

**SET** node after tail to new node

**SET** tail to new node

**INCREMENT** size

The prepend method, which takes bid as parameter and adds a new node to the beginning of the linked list. First we create a new node pointer and pass bid as a parameter which calls the constructor Node(bid aBid). Next an if statement check if the head is null, if it is, head and tail is set equal to new node, meaning the list is empty. If the head isn’t null, then the new node next is set to the head and the head is updated to new node. Finally, size is increased by 1.

**Prepend Pseudocode**

**CREATE** new node

**IF** head is null

**SET** head and tail to new node

**ELSE**

**SET** node after new node to the head

**SET** head to new node

**INCREMENT** size

The next fix-me is a print method which prints all bids, with the bid Id, title, amount, and fund. To do this a new node named current node is created and is set to the head. This node is used to traverse the linked list. Next a while loop is used to traverse the list and print each bid. Finally, current node is set to the next element in the list.

**Print list Pseudocode**

**CREATE** new node

**SET** new node to head

**WHILE** the new node isn’t null

**PRINT** bid

**SET** new node to new node next

The remove method takes bid ID as an argument and removes a bid that has a matching ID. First a check is performed to ensure the first node isn’t equal to the bid ID. If it is the head is set tom the next node and size is decreased by 1. If the head node doesn’t equal the bid ID, the list is traversed until it is found. If it’s found the next node is held onto by a temporary node and the current node (the one used to traverse the list) next node is set to the next node after the temporary node. But if the temporary node is equal to the tail, then the tail is set equal to the current node. Then the temporary node is deleted, and size is decreased by 1.

**Remove pseudocode**

**IF** head bids ID is equal to bid ID

The head is set to the next node after the head

**DECREMENT** size

**RETURN**

**SET** current node to head

**SET** temporary node to null

**WHILE** current node isn’t null

**IF** the next node after current node is equal to bid ID

**SET** temporary node to the next node after current

**SET** the next node after current node to the next node after the temporary node

**IF** the temporary node is equal to the tail

**SET** tail to current node

**DELETE** temporary node

**DECREMENT** size

**RETURN**

**SET** Current node equals the next node after current node

The search method searches for a bid ID and after it is found it is returned, if it isn’t found, a new empty bid is created and that is returned.

S**earch pseudocode**

**CHECK** if head is not null and head matches the bid ID

**IF it does**

**RETURN** head node

**CREATE** new node to traverse list set equal to head

**WHILE** current node isn’t null

**IF** the new node is equal to bid ID

**RETURN** new node bid info

**SET** new node to the next node after new node

**IF** no matching bid ID

**CREATE** empty bid

**RETURN** empty bid