**Module 4 Reflection and Pseudocode**

Jermaine Wiggins

Southern New Hampshire University

CS300 Analysis and Design

Prof. Sathish Gopalakrishnan

Feb 2, 2025

**Module 4 Reflection and Pseudocode**

This week’s assignment was challenging as I am still struggling with linked list, but I understand the overall concept of using them to overcome collisions when adding data to a hash table.

The purpose of this program is to load a collection of bids into a hash table, from there the user can display all bids, find a bid, or delete a bid.

The first fix-me was to initialize node structure by resizing the hash table in the default constructor, which resizes the node vector to match table size which is set to DEFAULT\_SIZE, if an argument isn’t given. If an argument is given, we resize node vector to argument.

**Default Constructor Pseudocode**

**RESIZE** nodes vector to match table size

**Constructor with argument Pseudocode**

**SET** table size to argument

**RESIZE** nodes vector to match table size

The next fix me was to free storage when class is destroyed, so we simply erase the node at the beginning.

**Destructor Pseudocode**

**ERASE** node at the beginning

Next, we must calculate a hash value, so we take the key argument and use modulus table size.

**Hash key Pseudocode**

**RETURN** key % tablesize

The insert method inserts a new bid into table and handles collisions by adding new bids with the same key to a linked list at that node.

**Insert Pseudocode**

**CREATE** new key

**GET** node that corresponds to key

**IF** node not found

**INSERT** into table

**ELSE** there is a collision, node exist at key

**FIND** the next open node

**INSERT** into table

The print all method simply prints all elements in the hash table.

**Print all Pseudocode**

**FOR** each index i in the hash table

**CREATE** new node(current) set to node at index i

**IF** current node key not empty

**PRINT** bid info (output key, bid ID, title, amount and fund)

**WHILE** node after current not null

**SET** current node to next node

**PRINT** bid info (output key, bid ID, title, amount and fund)

Next, we implement the remove function, which removes a bid from the hash table based on bid ID. It handles deletion at the head middle and if it’s the only bid in table differently.

**Remove Pseudocode**

**CREATE** new key

**CREATE** new node(current) and set to node at key

**CREATE** new node(previous) set to null

**WHILE** the current node isn’t null

**IF** current nodes bid ID is equal to bidID

**IF** previous node is null

**SET** previous to next node after current

**SET** current to previous

**DELETE** previous and r **RETURN**

**ELSE IF** isn’t equal to null

**SET** next node after previous to currents next node

**DELETE** current and **RETURN**

**ELSE**

Reset bid data to default/empty

**MOVE** to next node

**IF** bid not found **PRINT** error message

The search method searches the hash table for the bid that matched bid ID, when found it prints the bid info (output key, bid ID, title, amount and fund)

**Search Pseudocode**

**CREATE** new key

**CREATE** new node(current) and set to node at key

**WHILE** the current node isn’t null

**IF** current nodes bid ID is equal to bidID

**RETURN** bid info

**MOVE** to next node

**IF** not found **RETURN** emptybid