**Seema Chavesta**

Sep 28, 2024.

**Code Reflection:**

The purpose of this code is to implement a hash table for managing bids in an auction system using a unique bidId as the key. The core techniques include a hash function that maps bid IDs to indices in the table and chaining to handle collisions, where multiple bids hashing to the same index are stored in a linked list. Insertion adds bids to the appropriate index, searching traverses the list to find a specific bid, and removal updates the linked list by deleting the matching bid. Key challenges addressed include handling collisions efficiently through chaining and ensuring proper memory management with dynamic allocation. The code balances performance and reliability by distributing bids evenly across the table and managing linked list structures carefully.

**Pseudocode:**

1. Start program

2. Initialize HashTable with a default or user-defined size

3. Present user menu with options:

1. Load Bids

2. Display All Bids

3. Find Bid

4. Remove Bid

9. Exit

4. For each option:

- If "Load Bids" selected:

1. Parse CSV file and read each bid.

2. Use hash function to calculate index for each bid.

3. Insert each bid into the hash table.

- If "Display All Bids" selected:

1. Traverse hash table and print all stored bids.

- If "Find Bid" selected:

1. Get bidId input.

2. Calculate hash, search the linked list at the hashed index.

3. If found, display bid details; if not, display "not found."

- If "Remove Bid" selected:

1. Get bidId input.

2. Calculate hash, find bid in linked list, and remove it.

5. Loop until the user selects "Exit".

6. End program.