Ryan Hatch  
HashTable - Reflection

The HashTable program is a program that is meant to manage a collection of bids via a hash table, utilizing chaining to address collisions. This structure proves highly effective for operations such as reading, searching, and removing elements, which is vital for handling large datasets like those of a municipal government's auction bid records.

The source to this project is within the HashTable class, which oversees a vector of Node pointers. Each Node holds bid data and a key derived from a hash function. The class's main methods are Insert, PrintAll, Remove, and Search.

- **Load**: Opens a CSV file containing bids, reads the bids, and then uses "Insert" in order to place a new bid into the hash table, while handling collisions via chaining.

- **PrintAll**: Traverses each bucket and chain in the hash table to display all bids.

- **Remove**: Eliminates a bid based on a specified bid ID, ensuring the chain's integrity is preserved.

- **Search**: Retrieves and returns a bid that matches the given bid ID.

Developing this program served as a pretty good exercise for hands on learning on understanding and implementing hash tables. Personally, one big challenge was to ensure the hash function yielded an even distribution of keys to reduce collisions, along with effectively applying chaining. I also struggled with the implementation of the hash function and the nodes within the hash table. In order to address these challenges, I used a very systematic approach, simply debugging to track the program's flow and verify correct pointer management. I also had to make sure that the program had no problems with memory leaks or dangling pointers.

**Pseudocode:**

Program HashTable

Define DEFAULT\_SIZE as 179

Define GLOBAL\_SLEEP\_TIME as 5000

Structure Bid

Define bidId as String

Define title as String

Define fund as String

Define amount as Double

Structure Node

Define bid as Bid

Define key as Unsigned Integer

Define next as Node

Class HashTable

Private:

Declare nodes as Vector of Node Pointers

Declare tableSize as Unsigned Integer initialized to DEFAULT\_SIZE

Declare Hash Method accepting an Integer

Public:

Constructor HashTable()

Constructor HashTable(size)

Destructor ~HashTable()

Method Insert(bid)

Method PrintAll()

Method Remove(bidId)

Method Search(bidId)

Method Hash(bidId)

HashTable Constructor

Resize nodes to tableSize and initialize with nullptr

HashTable Destructor

Erase nodes from beginning to end

Method Hash(bidId)

Initialize hashValue to 0

For each character in bidId

Multiply hashValue by 31 and add ASCII value of character

Return hashValue modulo tableSize

Method Insert(bid)

Generate key using Hash(bidId)

If no node at key

Insert new Node at key

Else

Find the end of the list and attach new Node

Method PrintAll()

For each node in the table

Print bid information

Method Remove(bidId)

Generate key using Hash(bidId)

If bid is found at key

Remove bid and manage pointers

Method Search(bidId)

Generate key using Hash(bidId)

If bid is found at key

Return bid

Else

Return empty bid

Method main()

Process command line arguments for CSV file and bidKey

Initialize hash table

While user choice not equal to 'Exit'

Display menu options

Get user choice

Perform action based on choice:

Load Bids: Call loadBids() and measure time

Display All Bids: Call PrintAll()

Find Bid: Call Search(bidKey) and measure time

Remove Bid: Call Remove(bidKey)

Exit: End program

End While

End Program

End Program HashTable