// Define a structure to hold bid information

Structure Bid:

String bidId

String title

String fund

Double amount

Constructor:

amount = 0.0

// Define a class for Hash Table

Class HashTable:

// Inner class for Node

Class Node:

Bid bid

Unsigned int key

Node\* next

Constructor:

key = UINT\_MAX

next = nullptr

Constructor with Bid:

Call default constructor

bid = input bid

Constructor with Bid and key:

Call constructor with Bid

key = input key

// Private members

Vector of Node pointers nodes

Unsigned int tableSize = DEFAULT\_SIZE

// Private method to calculate hash

Unsigned int hash(int key):

Return key modulo tableSize

// Public methods

Constructor:

Resize nodes to tableSize, initialize with nullptr

Constructor with size:

Set tableSize to input size

Resize nodes to tableSize, initialize with nullptr

Destructor:

For each node in nodes:

While node is not null:

Store current node in temp

Move to next node

Delete temp

Method Insert(Bid bid):

key = hash(atoi(bid.bidId.c\_str()))

Set prev and current to nullptr and nodes[key] respectively

If current is null:

Create new Node with bid and key, assign to nodes[key]

Else:

While current is not null:

If current key is UINT\_MAX:

Set current key to key, bid to input bid, next to nullptr

Return

Set prev to current, move to next node

Create new Node with bid and key, assign to prev next

Method PrintAll():

For each node in nodes:

While node is not null and node key is not UINT\_MAX:

Print node key, bidID, title, amount, and fund

Move to next node

Method Remove(String bidId):

key = hash(atoi(bidId.c\_str()))

Set current and prev to nodes[key] and nullptr

While current is not null and current bidId is not equal to bidId:

Set prev to current, move to next node

If current is null, return

If prev is null:

Set nodes[key] to current next

Else:

Set prev next to current next

Delete current

Method Search(String bidId):

Create an empty Bid object

key = hash(atoi(bidId.c\_str()))

Set current to nodes[key]

While current is not null:

If current bidId is equal to bidId:

Return current bid

Move to next node

Return empty bid

// Static method to display bid information

Function displayBid(Bid bid):

Print bidId, title, amount, and fund

// Static method to load bids from CSV file

Function loadBids(String csvPath, HashTable\* hashTable):

Print loading message with csvPath

Initialize CSV parser with csvPath

Print CSV header

Try:

For each row in CSV file:

Create a Bid object

Set bidId, title, fund, and amount from CSV row

Insert bid into hashTable

Catch CSV error:

Print error message

// Static method to convert string to double

Function strToDouble(String str, char ch):

Remove all occurrences of ch from str

Return atof(str.c\_str())

// Main function

Function main(int argc, char\* argv[]):

Parse command line arguments

Define timer variable

Create a new HashTable object

Initialize bid and choice

While choice is not 9:

Print menu options

Get user input for choice

Switch on choice:

Case 1:

Start timer

Call loadBids with csvPath and bidTable

Calculate and print elapsed time

Case 2:

Call PrintAll on bidTable

Case 3:

Start timer

Call Search on bidTable with bidKey

Calculate and print elapsed time

If bidId is not empty:

Call displayBid with bid

Else:

Print bid not found message

Case 4:

Call Remove on bidTable with bidKey

Print goodbye message

Return 0