Hash Table Pseudocode

Dakota Keyes

CS300

//Define a structure to hold bid info

struct Bid {

string bidId; // unique identifier

string title;

string fund;

//Class w/ data members and methods to implement

class HashTable {

struct Node {

Bid bid;

unsigned int key;

Node \*next;

//Default constructor

key = UINT\_MAX;

next = nullptr;

// initialize with a bid and a key

Node(Bid aBid, unsigned int aKey) : Node(aBid) {

key = aKey;

//Structure to Hold bids

HashTable::HashTable() {

nodes.resize(tableSize);

}

// free storage

nodes.erase(nodes.begin());

//Calculate hash value

return key % tableSize;

void HashTable::Insert(Bid bid) {

//Inserting a bid & creating key for the given bid

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

Node\* oldNode = &(nodes.at(key));

// retrieving the node using key

if (oldNode == nullptr) {

Node\* newNode = new Node(bid, key);

nodes.insert(nodes.begin() + key, (\*newNode));

}

else {

if (oldNode->key == UINT\_MAX) {

oldNode->key = key; ( oldNode->bid = bid) (oldNode->next = nullptr) }

else {

while (oldNode->next != nullptr) {

oldNode = oldNode->next; }

oldNode->next = new Node(bid, key); }

//For loop to print

void HashTable::PrintAll() {

for (auto it = nodes.begin(); it != nodes.end(); ++it) {

if (it->key != UINT\_MAX) {

PRINT “Key”>> key, bid.bidId >> bid.title >>

void HashTable::Remove(string bidId) {

//for loop to remove a bid

//Searching for a bid

Bid HashTable::Search(string bidId) {

Bid bid;

If(entry found)

Return key;

If(no entry found)

Return bid;

While(node!=null ptr)

If(current node matches)

Return;

Node = next->node;

Return bid;