Structure Node

Bid bid

Node\* left

Node\* right

Structure Bid

string bidId

string title

string fund

double amount

Class BinarySearchTree

Node\* root

// Constructor

Function BinarySearchTree()

root = nullptr

// Destructor

Function ~BinarySearchTree()

// Recurse from root deleting every node

// Insert a bid

Function Insert(Bid bid)

If root is null

root = new Node(bid)

Else

addNode(root, bid)

// Add a bid to some node (recursive)

Function addNode(Node\* node, Bid bid)

If bidId < node->bid.bidId

If node->left is null

node->left = new Node(bid)

Else

addNode(node->left, bid)

Else

If node->right is null

node->right = new Node(bid)

Else

addNode(node->right, bid)

// Remove a bid

Function Remove(string bidId)

root = removeNode(root, bidId)

// Remove a bid from some node (recursive)

Function removeNode(Node\* node, string bidId)

If node is null

return node

If bidId < node->bid.bidId

node->left = removeNode(node->left, bidId)

Else If bidId > node->bid.bidId

node->right = removeNode(node->right, bidId)

Else

If node->left is null

Node\* temp = node->right

delete node

return temp

Else If node->right is null

Node\* temp = node->left

delete node

return temp

Node\* temp = node->right

While temp->left is not null

temp = temp->left

node->bid = temp->bid

node->right = removeNode(node->right, temp->bid.bidId)

return node

// Search for a bid

Function Search(string bidId)

Node\* current = root

While current is not null

If bidId == current->bid.bidId

return current->bid

If bidId < current->bid.bidId

current = current->left

Else

current = current->right

Bid bid

return bid

// Traverse the tree in order

Function InOrder()

inOrder(root)

Function inOrder(Node\* node)

If node is not null

inOrder(node->left)

displayBid(node->bid)

inOrder(node->right)

// Traverse the tree in post-order

Function PostOrder()

postOrder(root)

Function postOrder(Node\* node)

If node is not null

postOrder(node->left)

postOrder(node->right)

displayBid(node->bid)

// Traverse the tree in pre-order

Function PreOrder()

preOrder(root)

Function preOrder(Node\* node)

If node is not null

displayBid(node->bid)

preOrder(node->left)

preOrder(node->right)

// Function to display the bid information to the console

Function displayBid(Bid bid)

Print bid.bidId, bid.title, bid.amount, bid.fund

// Function to load a CSV file containing bids into a container

Function loadBids(string csvPath, BinarySearchTree\* bst)

Print "Loading CSV file", csvPath

Initialize CSV parser with csvPath

For each row in CSV file

Create a Bid

Set bidId, title, fund, amount

Insert bid into bst

// Function to convert a string to a double after stripping out unwanted char

Function strToDouble(string str, char ch)

Remove char ch from str

Return atof(str)

// Main function

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

Parse command line arguments for csvPath and bidKey

Initialize a timer variable

Create a BinarySearchTree bst

Create a Bid bid

While choice != 9

Print menu options

Get user choice

If choice == 1

Start timer

Load bids from CSV into bst

Print elapsed time

Else If choice == 2

bst.InOrder()

Else If choice == 3

Start timer

bid = bst.Search(bidKey)

Print elapsed time

If bid is found

Display bid

Else

Print "Bid Id not found"

Else If choice == 4

bst.Remove(bidKey)

Print "Good bye."