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

**Define a binary search tree to hold all bids**

**Initialize a variable called “choice” of type int and set it equal to 0**

**While choice does not equal 9**

**Display a menu prompting the user to “1. Load Bids”, “2. Display Bids”, “3. Find Bid”, “9. Exit”**

**In the case the user selects 1**

**Set the variable “ticks” equal to the “clock()” function**

**Use the “loadBids()” function with the user’s input to pass in the**  **parameters**

**Set the variable “ticks” equal to the clock() function – “ticks"**

**Output the time in clock ticks**

**Output the time in seconds**

**Then make a break in the case**

**In the case the user selects 2**

**Use the bst’s object pointer to call the “InOrder()” function to display**  **all courses in alphanumeric order**

**Then make a break in the case**

**In the case the user selects 3**

**Set the variable “ticks” equal to the “clock()” function**

**Set the “bid” variable equal to the “bst” object’s pointer to the**  **“search()” function where bidKey is passed in as the parameter**

**Set the variable “ticks” equal to the clock() function – “ticks"**

**If the bid Id is not empty**

**Use the “displayBid()” function and pass in the “bid” variable**

**Else**

**Output: “Bid Id is not found”**

**Output the time in clock ticks**

**Output the time in seconds**

**Then make a break in the case**

**In the case the user selects 9**

**Output: “Good bye”**

**End the program**

}

void BinarySearchTree::InOrder() {

**Call the “inOrder” function and pass in the root**

}

void BinarySearchTree::inOrder (Node\* node) {

**If node is equal to a null pointer**

**Return void**

**Use the “inOrder” function and pass in the left node**

**Output the bidID, title, amount and fund**

**Use the “inOrder” function and pass in the right node**

}

void VectorSort::selectionSort() {

**Set a variable called “min” with type int to 0**

**Define an un-initialized “temp” variable with a string type**

**If the size of the bids vector is equal to 0**

**Then return null**

**Set a variable called “size” with a type of size\_t equal to the size of the bids vector.**

**Set a variable called “pos” with type int to 0**

**For i = 0 and is less than the size of the vector minus 1**

**Set the “min” variable to the “pos” variable**

**For j = 1 + i and is less than the size of the vector**

**If the element’s title is less than the minimum title**

**“Min” becomes equal to the element of j**

**Swap the current minimum with the smaller one found in the for loop**

}

void HashTable::PrintAll () {

**For int “i” is equal to 0 and is less than the size of the nodes**

**Output the nodes’ key, bidId, title, amount, and fund**

**Set a variable called “current” of type Node\* equal to nodes[i].next**

**While the “current” Node\* variable is not equal to a null pointer**

**Output the nodes’ key, bidId, title, amount, and fund from the chain**

**“current” will equal the next node in the list**

}