# Thomas Martin

# CS-300-H2990 DSA: Analysis and Design 23EW2

# 5-2 Assignment: Binary Search Tree

# Southern New Hampshire University

**December 02, 2023**

**Code Reflection**

The code is designed to implement a basic Binary Search Tree. It includes structures for bids and nodes, and a class BinarySearchTree with methods for inserting, removing, searching, and displaying bids. The purpose is to manage bids efficiently using a binary search tree data structure. Focusing on modularizing the code, ensuring readability, and using appropriate variable names. I encountered challenges in implementing the removal operation, especially dealing with nodes having two children.

**Pseudocode**

Node {

Bid bid;

Node\* left;

Node\* right;

Node(Bid \_bid) {

bid = \_bid;

left = nullptr;

right = nullptr;

}

}

class BinarySearchTree {

private:

Node\* root;

void addNode(Node\* node, Bid bid);

void inOrder(Node\* node);

Node\* removeNode(Node\* node, string bidId);

public:

BinarySearchTree();

virtual ~BinarySearchTree();

void InOrder();

void Insert(Bid bid);

void Remove(string bidId);

Bid Search(string bidId);

void addNode(Node\* node, Bid bid); // Duplicate declaration?

void inOrder(Node\* node); // Duplicate declaration?

Node\* removeNode(Node\* node, string bidId); // Duplicate declaration?

}

BinarySearchTree\* bst = new BinarySearchTree();

// Load bids into the binary search tree

loadBids(csvPath, bst);

// Display the bid information

Bid bid = bst->Search(bidKey);

if (!bid.bidId.empty()) {

cout << "Bid Information:" << endl;

displayBid(bid);

} else {

cout << "Bid not found." << endl;

}

**Specifications and Correctness**

The class should correctly implement the Binary Search Tree operations: insert, remove, search, and display. It should handle edge cases, such as an empty tree or removing a node with two children.

**Annotation / Documentation**

The code should be well-commented to explain the purpose of lines or sections of code. Special attention should be given to complex algorithms and error-handling sections.

**Modular and Reusable**

Methods within the BinarySearchTree class should have distinct responsibilities (e.g., one method for inserting, another for removing). Methods should be designed to be reusable, adhering to the single responsibility principle.

**Readability**

The code should have a consistent indentation and appropriate use of blank lines to separate distinct parts of the code and operations. Whitespace should be used consistently. Follow a consistent naming convention (e.g., camelCase or underscored), ensuring clarity.