Devin wheeler

4-2 Milestone Three: Enhancement Two: Algorithms and Data Structure

**Introduction**

This artifact is a Binary Search Tree (BST) program that is written in C++. It was originally developed for CS 300: Data Structures and Algorithms. It was made to show off the abilities to use a BST for insertion, deletion, and in order traversal. However, I wanted to add a way to filter data and then sort it. So, I added the ability to soft by less than a winning bid amount and then filter it by either data of department.

**Justification**

I selected this artifact to include in my portfolio because it’s a great way to show my understanding of algorithms and data structures. The enhanced version not only shows off my ability to work with trees but also my ability to add algorithmic enhancements like data filtering, sorting, and input validation. These enhancements show off my knowledge in search algorithms, tree traversal, data filtering, and sorting algos. Which are all great skills to have in software development.

The enhancements include:

* Allowing users to filter bids by max winning amount
* Adding the ability to sort the filtered data by either department that one the bid or date it was won.
* Adding validation to user inputs
* Adding a dateToInt() algorithm to sort by date accurately and not just by the first number.
* Cleaning up and adding needed comments.

These updates add to the useability of the program while keeping it efficient, showing my skills in managing time and space complexity.

**Outcomes**

These improvements align with the Computer Science program outcomes by showing my ability to:

* **Algorithmic Solutions:** Used algorithmic principles to implement custom filtering functionality using an in-order BST traversal (O(n)) and sorting method based on user input (O(m log m)).
* **Technical Implementation:** Showed proficiency in C++ by extending the functionality of a BST by adding filtering, sorting, and date-handling logic using custom solutions like dateToInt().
* **Professional Communication:** Improved code readability and usability by adding user input validation, helpful prompts, and cleaning comments to clarify the functionality of the code and ensure the application is easy to use and understand for everyone.

**Reflection**

Enhancing this artifact taught me a lot about balancing usability with efficiency. One of the hardest challenges I faced was sorting the string formatted dates. At first, I couldn’t figure out why my date sort was not working currently. After looking at it for some time I realized C++ was just doing a string compare. So, I decided to break the string apart and put it back together as an integer that would be sortable. This was a reminder of how important data formatting and data processing can be.