The artifact I selected for Category Two is a C++ program that implements a binary search tree (BST) to manage a collection of college courses. Each course is stored as a node in the tree and includes fields for the course number, course name, and any prerequisites. The program allows users to load course data from a file, display an alphabetical list of courses, and search for specific course details. This artifact was originally created during a previous data structures course as part of an assignment focused on BST operations such as insertion, traversal, and search.

I selected this artifact because it demonstrates my understanding of recursive algorithms and the application of binary search trees which are foundational data structures in computer science. It highlights my ability to work with pointer-based data structures, file input/output, and menu-driven logic in a C++ environment. The enhancement I made to this artifact was the addition of a recursive function that counts the number of nodes or “courses” in the BST. I also added a new option to the main menu that allows the user to choose this function and display the total course count. This modification improved the artifact by adding a meaningful feature that further exercises recursive tree traversal, enhances usability, and reflects my growing confidence in manipulating existing codebases.

Yes, I met the outcomes I originally planned to address in Module One. This enhancement specifically supports the outcome of designing and evaluating computing solutions using algorithmic principles and data structures. The recursive ‘countCourses()’ function demonstrates my ability to reason with tree-based recursion and apply it to real world problems. I do not have any updates to my outcome-coverage plans and this enhancement remains aligned with my original intent and supports progress toward mastery in data structures and algorithms.

Enhancing the artifact required me to revisit my understanding of recursion and tree traversal. I was already familiar with in order traversal from the original assignment so implementing a function to count all nodes helped reinforce how traversal logic can be adapted to solve different problems. One of the challenges I faced was making sure that the count function worked correctly even when the tree was empty or had only one node. I also had to carefully integrate the new feature into the existing menu system without disrupting other program functionality. Through this process, I gained more confidence in my ability to refactor and expand C++ code, particularly in applying abstraction to write recursive solutions.