Patrick Gallagher

November 3, 2025

Project 3 Report

LLM Usage

For Project 3, I did not use any AI assistance. The following prompts are ways that I could have used AI.

Prompts

1. “Please convert my current method to return a bool typing instead of void. Wherever a recursive call happens, there should be a returned Boolean. All of the returned recursive calls in each call should be OR-ed together to return a single Boolean. Also, return true if the last value is located and the key statement is printed.”
    a. During development, I found a moment where I had fully developed the find method without the consideration for the EMPTY statement. To amend this, I spent almost an hour troubleshooting and rewriting the function. This task may have been accelerated through LLM assistance.
2. “My code is not compiling but my IDE is not throwing any errors. Please describe the area of my code that is causing the compiler to fail.”
    a. There were some moments during development that I could not compile the code due to some small syntax errors that were not throwing errors in VS Code. Rather than combing through the code myself, I could have asked for some help from an AI model.
3. “This is a sample input file for the NestedBST program and an explanation of the file format. Please develop a series of test inputs to examine special and edge cases. Also, create a modified version of my code that outputs a Binary Tree visualization of the input file.”
    a. Creating test inputs that examine all edge cases can be very time consuming, and possible source of human error. As a large database of programming knowledge, the AI model might be able to rapidly generate a robust series of test cases. I also experienced some difficulty in developing an understanding of the structure of the Nested Tree. An LLM implementation may have been able to assist in building my understanding.

Debug/Testing

Debugging

While debugging, I ran into a collection of relatively small issues. Once I began testing, my code ran into overflow errors, which resulted in crashes. After a brief investigation, I discovered that I was passing the (i++) statement into a function call, which was resulting in undesirable behavior. I quickly corrected this issue by changing the argument to (i + 1), which resulted in an almost correct output file, with the exception of an unexpected amount of display() calls. I checked the

main function and discovered that I had at some point deleted the break statement in the Find switch branch, which resulted in the display function being called with each find call. Another small issue that I encountered was in my print statements. In several of the for loops used to print the key path, I had forgotten to use the loop variable inside of the path vector, which resulted in the same number being printed 3 or 4 times rather than the proper path. The final error that I experienced in testing was the conversion of the find function from a void return function to a bool return function, as described in the AI section. After making these changes, my code passed all test inputs and the autograder tests.