Daniel Gregory

Search Trees

To get the program to compile I placed all the .csv files in the folder with the C++ code so the program can access the data easily. The majority of the program here are helper functions for constructing both trees as well as deleting in them. The binary search tree was fairly straightforward and not too bad to figure out, but the red-black tree was difficult with all the rules. I had ChatGPT help me with some of the rules for the red-black tree and then I had to go back and edit bits and pieces to match names but overall it was very helpful because I struggle with understanding red black trees. As far as the main function, it starts with opening the .csv file and grabs the information and calls the helper function to insert it into the tree. Then a second helper function is called to put the nodes in the order they belong in and print the tree. There is also a helper function called maxDepth to determine the max depth of each tree after insertion and deletion. The red-black search tree also follows the same pattern in the main function just with a ton of different helper functions based on all the rules for the red-black trees.

Lastly, the height of each tree or the max depth was what I expected, the red-black trees have a smaller height because it is a self balancing tree so it should be relatively small. The binary search tree had a much bigger height when using the bad data test set because it was in order, but the random set had about what I expected for the height. The height when deletion occurred was slightly smaller for the bad data in the binary search tree, but in the red-black tree the height was actually larger after deletion which I found really interesting.