Thomas Rafeld, M12808058

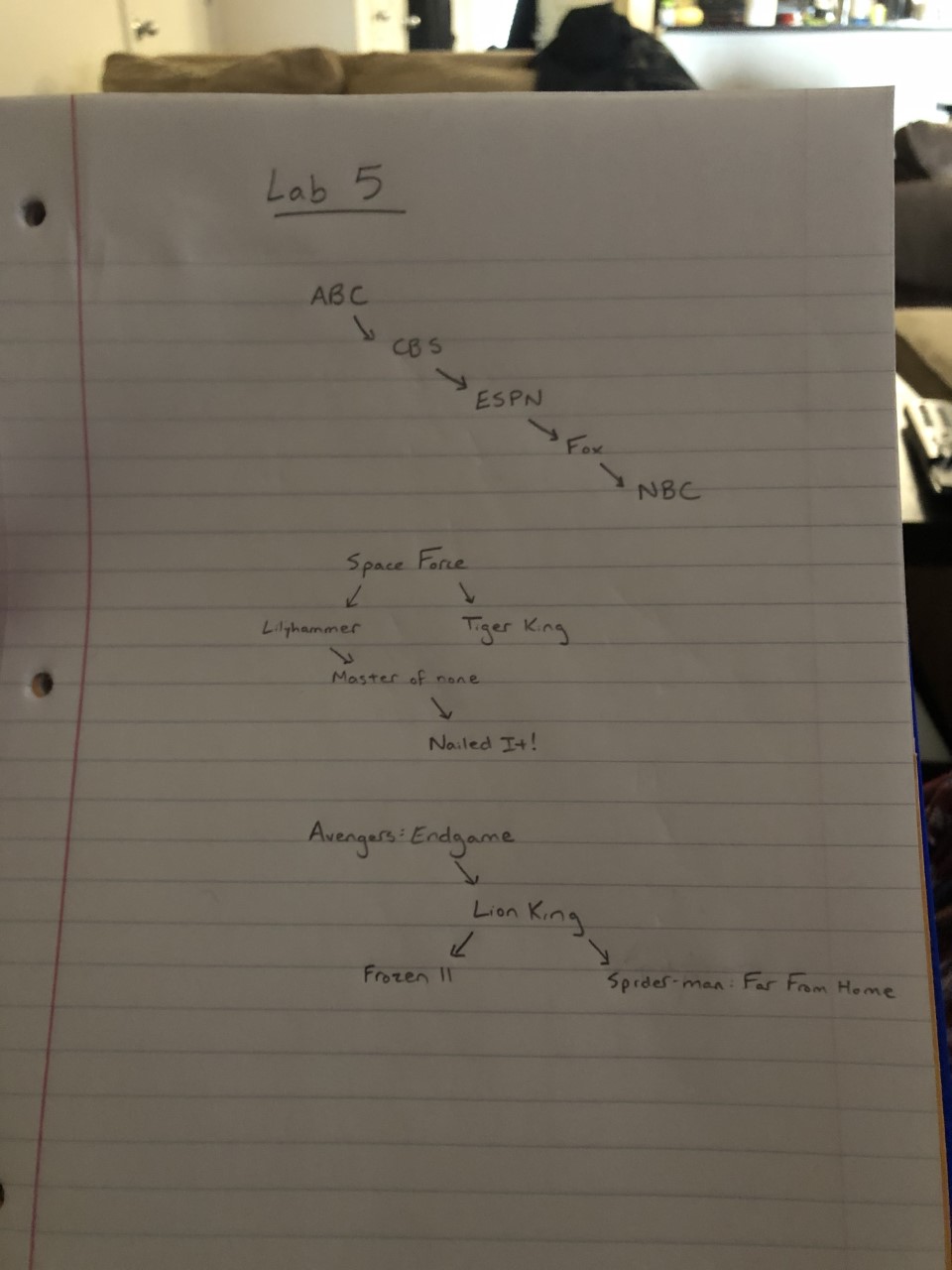
Ryan Winterhalter, M12727389

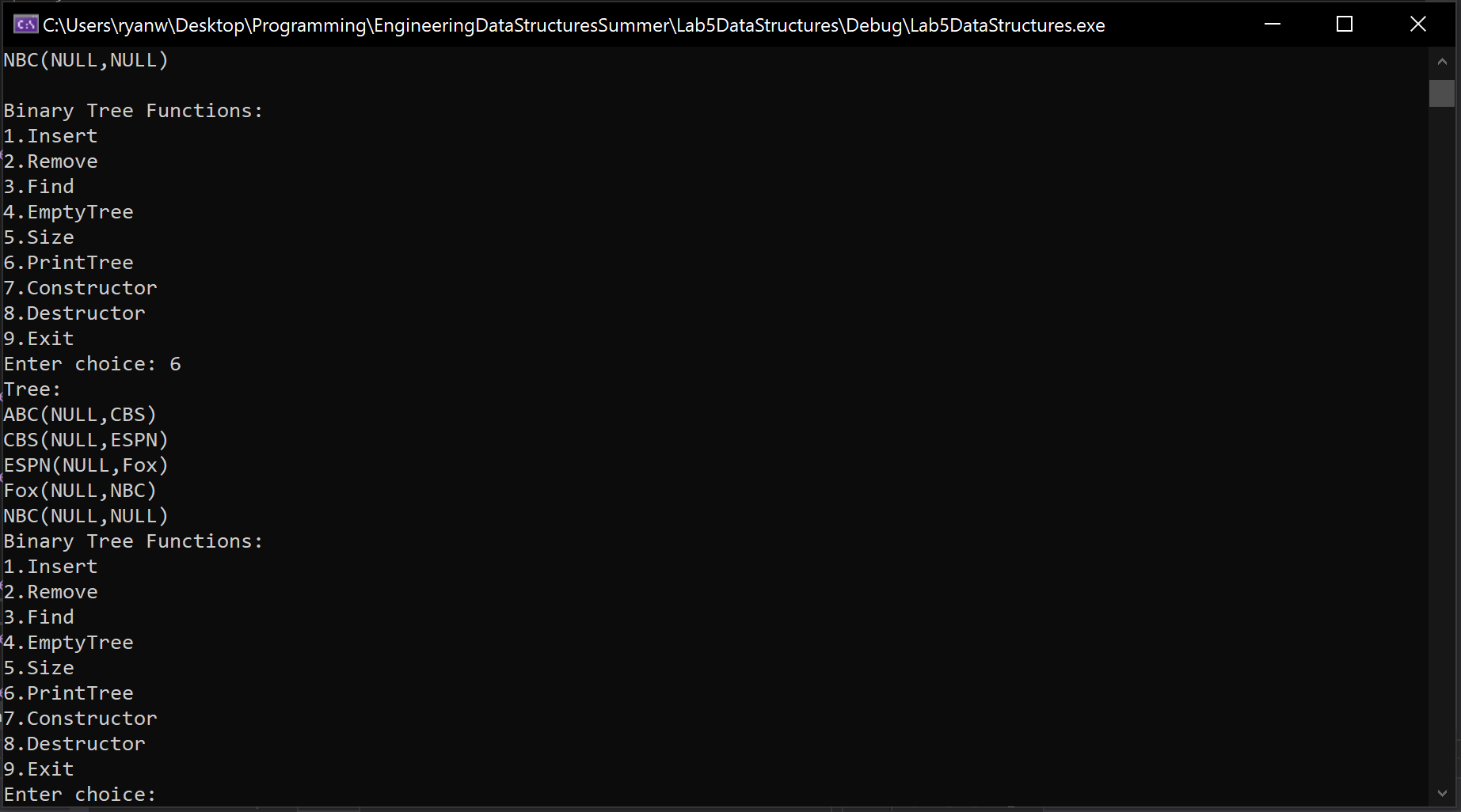
Lab 5 Report

**Objectives:**

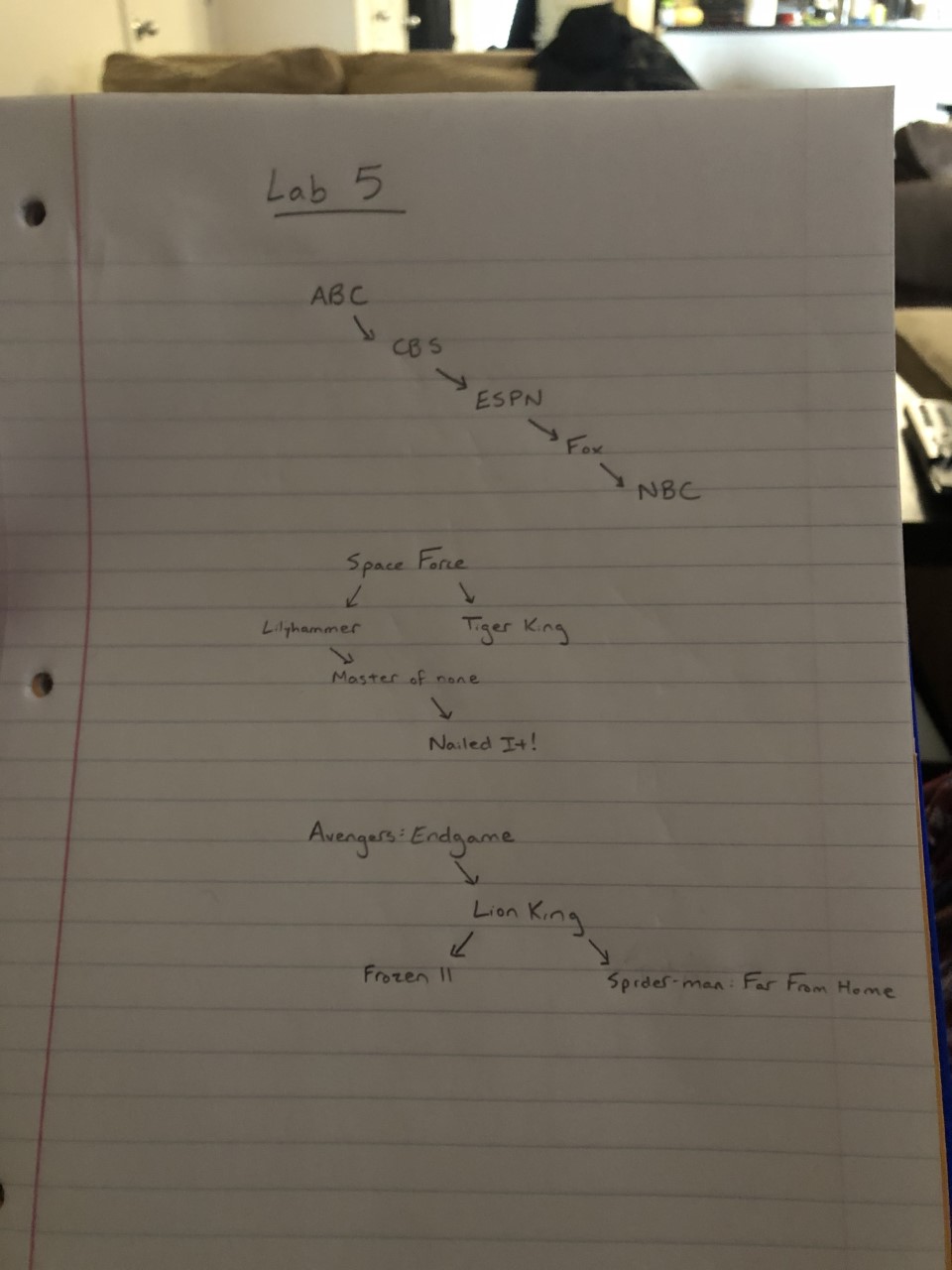
The objectives that we explored in this lab were the ability to understand efficiency as described by big O notation, understand binary trees, and build working binary trees. Understanding efficiency and big O notation is important to our success in this class because it allows us to understand how well our solutions to different problems work. We can use efficiency to optimize our solution, such as finding the one that requires the fewest operations or the one that uses the smallest amount of memory. Big O notation is an important way of describing the efficiency of our solutions in this class because it allows us to see how our solution would respond to a problem of a different size (such as increasing the amount of data we use within our structures). Similarly, this understanding of efficiency and big O notation is crucial to our success in engineering careers. In our careers, based on the specific project, there will be a set of criteria that our solution must meet and there will be some sort of preferred efficiency, such as programmer time, memory space, number of operations, etc. Understanding efficiency and the scope of the problem we are trying to solve is invaluable when it comes to choosing your desired solution so that you can generate a more efficient solution. Throughout this module, we demonstrated our understanding of efficiency and big O notation by watching the module video on efficiency and answering the discussion questions related to efficiency. The second objective of understanding binary trees is also incredibly important to our success in this class, as this module’s lab asked us to implement a binary tree. Binary trees are an important data structure for organizing data, so understanding what they are and how they work is very important to our success in this class. Because they are such an important data structure, understanding them is also important to our careers in engineering. When working on projects, we may want to use binary trees to implement our solution. Having an understanding of binary trees is therefore critical if we are to build or debug a solution that has a binary tree implementation. We demonstrated our understanding of binary trees throughout the entirety of this lab, as the lab required us to build and test a working binary tree. The third objective of being able to build working binary trees is important to our success in this class because this lab required us to build a functioning binary tree. It is also important to our success in engineering careers because, as engineers, the products we make need to work properly. Understanding what binary trees are and how they work is one thing, but being able to implement a working solution of a binary tree is crucial to a career in engineering as engineers are required to apply their understanding to generate real world solutions. We demonstrate our understanding of being able to build functioning binary trees throughout the entirety of lab 5, as the whole lab required us to construct a binary tree and its member functions that work as expected.

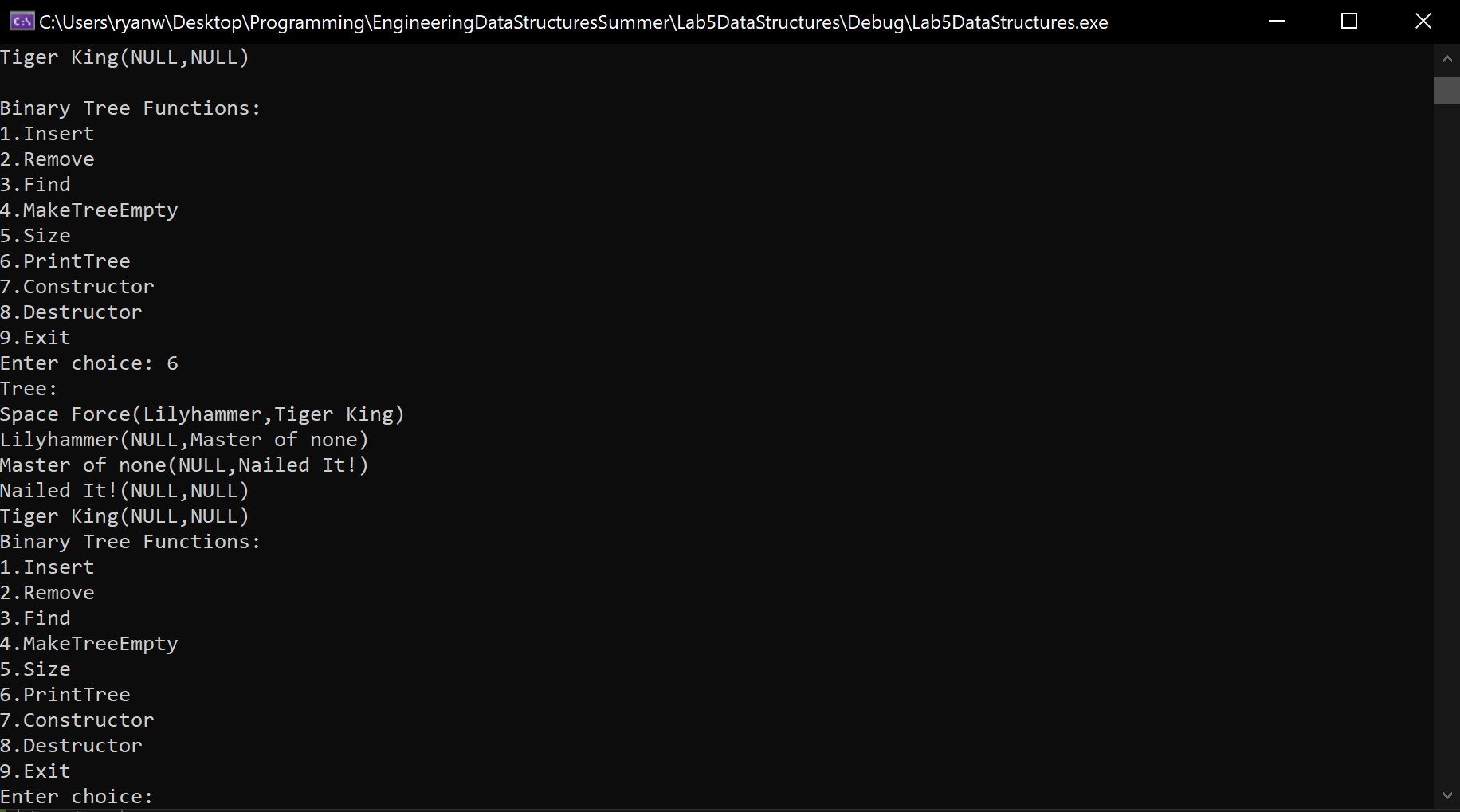
**Task 2a:**



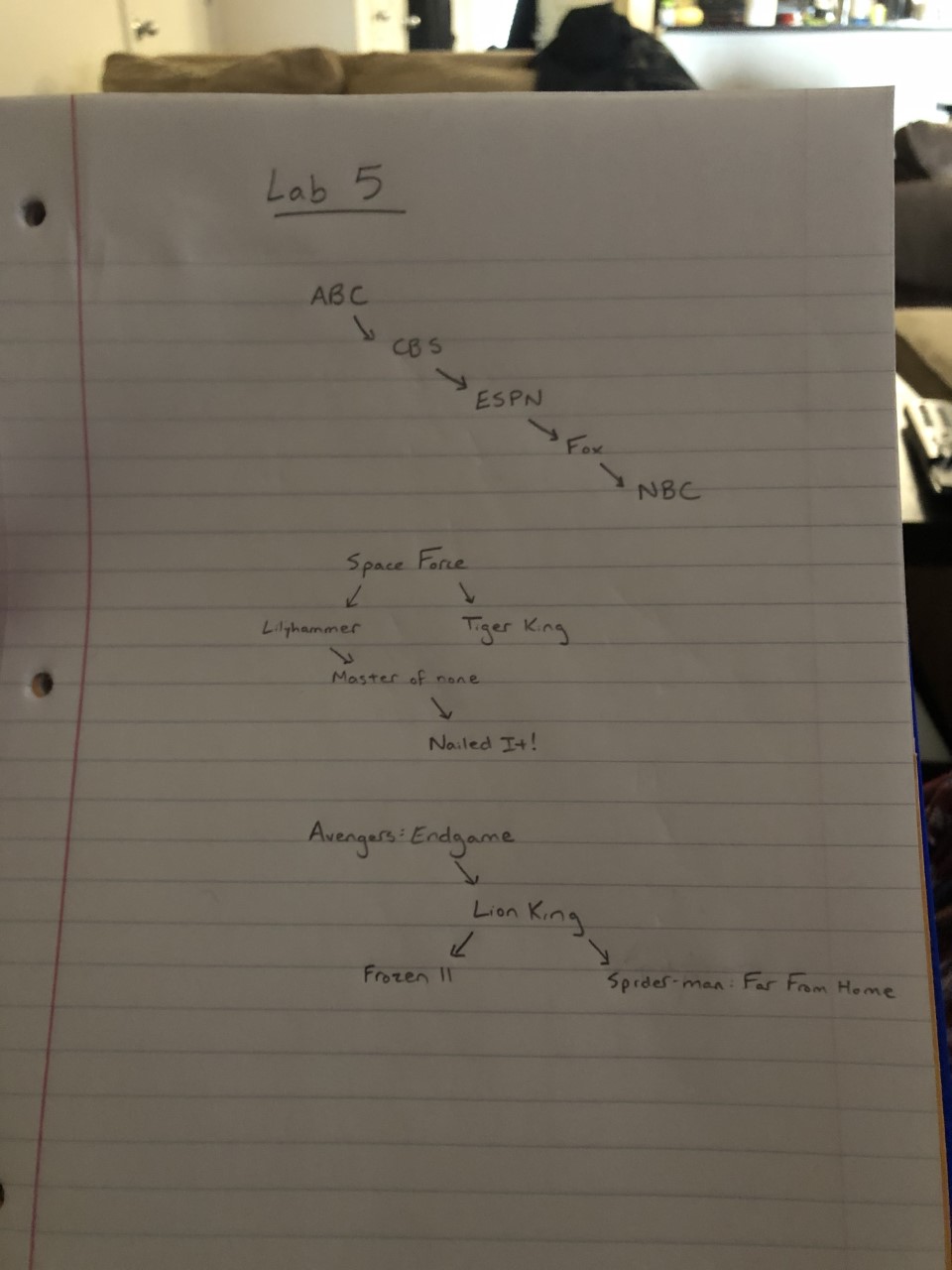


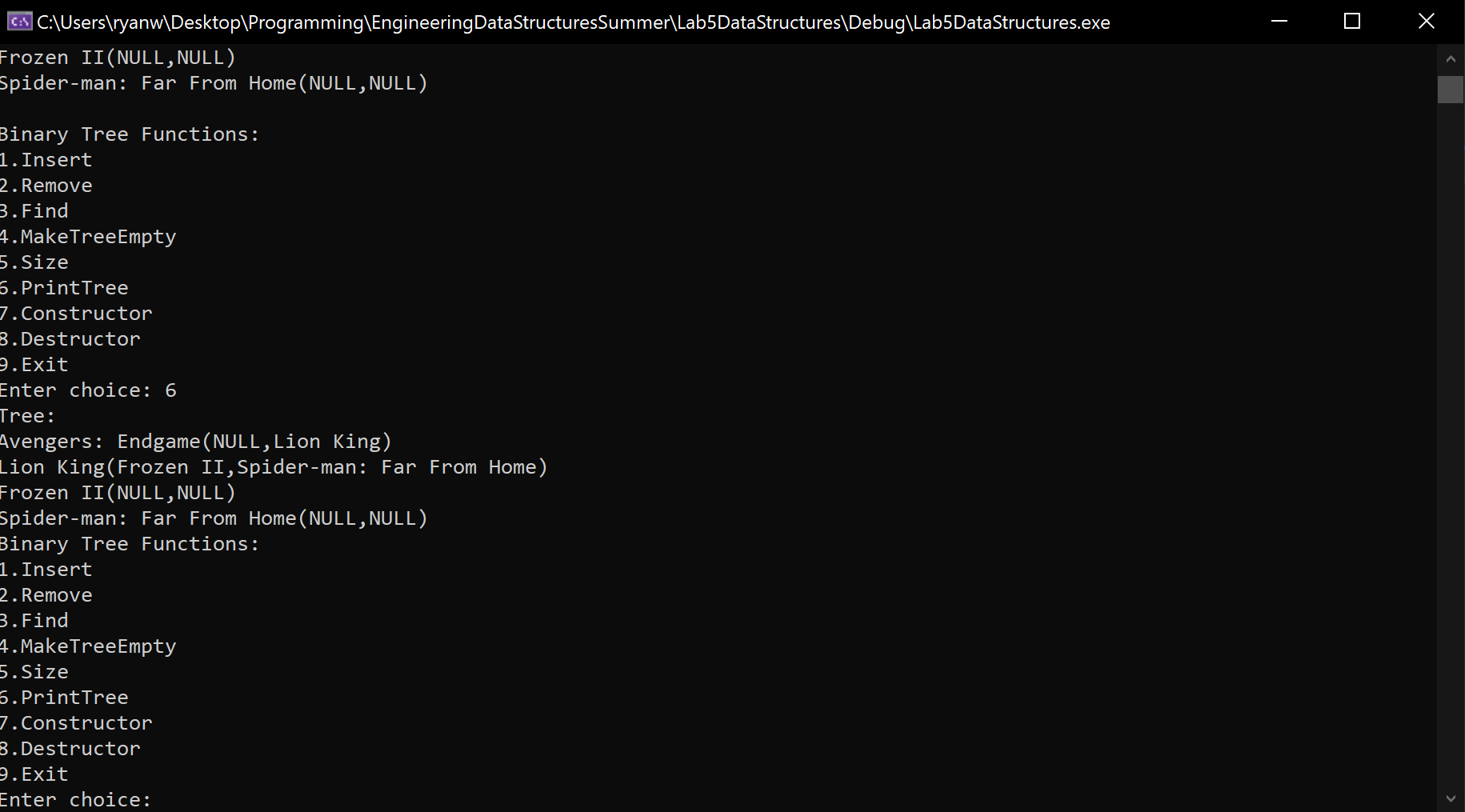
**Task 2b:**





**Task 2c:**





**Tree Structure:**

Our tree is structured to link values in the tree by their value: greater values will be stored to the right of smaller values and, likewise, smaller values will be stored on the left of greater values. The tree has this structure because it allows us to insert and find items in the tree very easily. When inserting and finding items, we can look at a node and know based on the value we would like to find or insert whether the next node we need to look at is to the left or to the right of the current node. We look through the binary tree in our code using pointers to the nodes in the tree. To improve the structure of our tree, we could modify it to be able to handle cases where the value we are trying to insert into the tree is already in the tree. This change would allow our tree to handle repeat values, which is something that our tree currently does not handle. However, this is a case that a certain implementation of a binary tree may require.

**Group Contributions:**

The lab was worked on together by both Ryan and Thomas while on a call together in Microsoft Teams. Both worked on creating the binary tree functions in the header file. Ryan wrote the code for removing nodes with two children, for the root, or just a normal node. Thomas wrote the main file that tests the functions of the binary tree class. In the end we used Ryan’s header file and the main testing file written by Thomas. For the final grade each member of the group should receive 100 percent of the grade as we feel that we both evenly contributed to the lab and worked together for almost the whole time it was being worked on.