Binary Search Tree Report

Donovan Byler – April 22, 2019

CS23001 – Computer Science 2

**Program Functionality**

This program creates a binary search tree using pointers with the numbers given via an input file. It is capable of inserting, deleting, sorting, and printing these variables. The program sorts and rearranges the tree whenever a value is removed from the tree.

**Design**

With this program, you can create large binary trees from a simple input of a file to an output in the form of a text-based description of the tree. The image below demonstrates the program’s capabilities as well as the display feature. The output is displayed on the console as well as in an output text file.

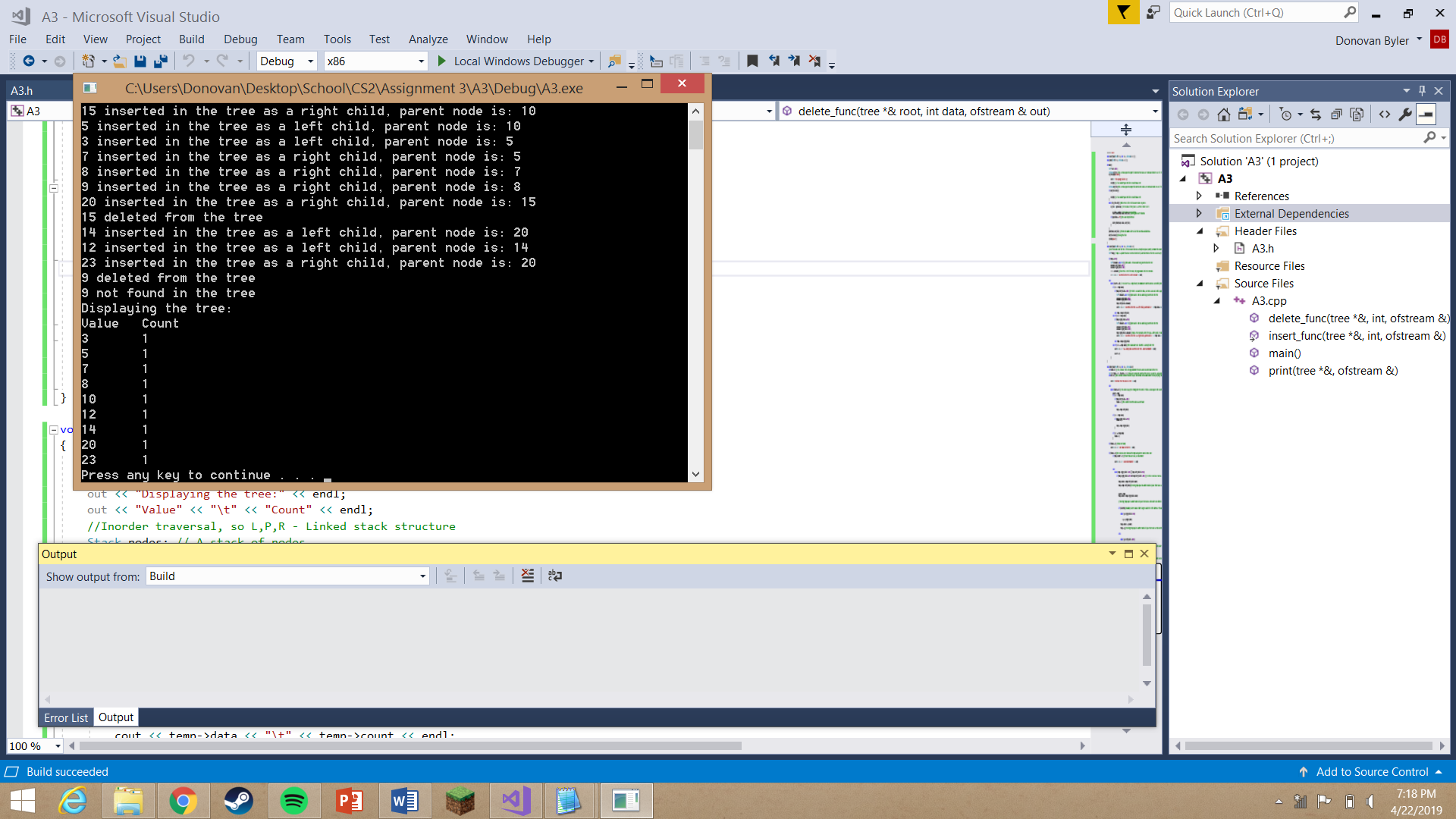


Figure 1: Output sample

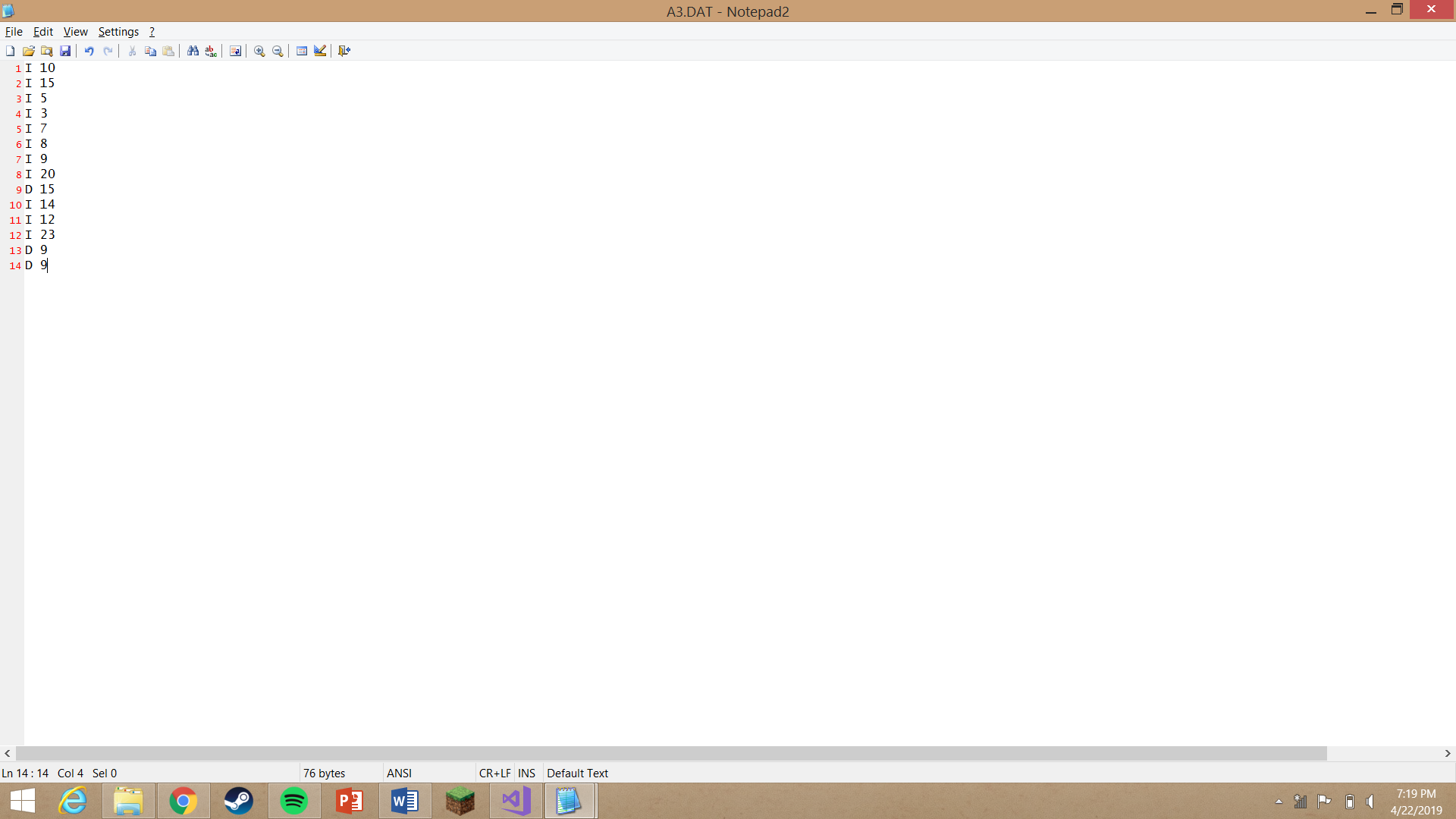


Figure 3: Input Sample

Alongside the driver function, the program has three simple functions: insert\_func, delete\_func, and print. Additionally, there is a struct which contains the data at each node as well as pointers to the left and right child nodes if applicable. The driver function opens the input and output files, reads the input which calls either a delete or insert function, prints the results, and finally, closes the files.

The insert function traverses through the nodes, moving through the right child of a node if larger or the left child if smaller. When it finds an empty node, the program creates a new node and places it in the empty spot. The system sends a report to the output indicating that the number was successfully placed as well as which node is its parent. If the value has already been placed, the count value of the placed node is simply incremented to note the increase.

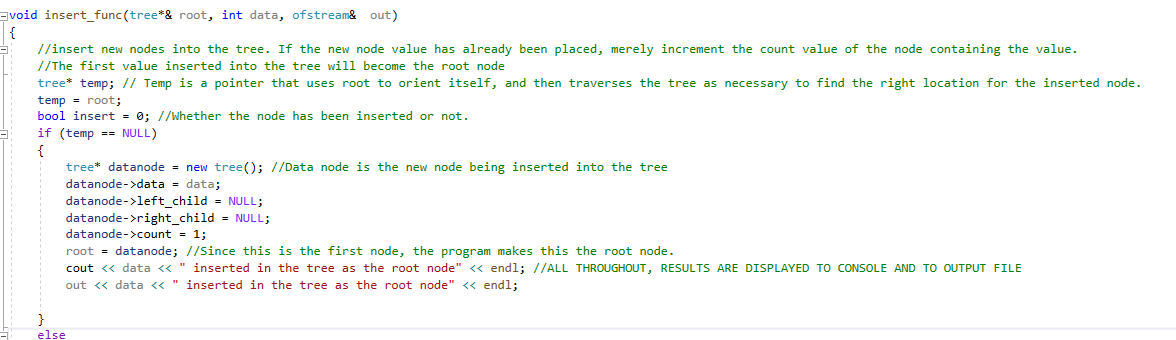


Figure 4: The beginning of the insert function

The delete function similarly passes through the nodes in search of the node that matches the value passed into the function (the node to be deleted). If the value is greater than the node, the program searches to the right. If smaller, it searches the left side. If the value is not found the program alerts the user that the value is not in the tree. Additionally, if the number has been added twice, the program de-increments the count rather than deleting the node. If the node to be deleted has child nodes, the program rearranges the tree, replacing the deleted node with either the largest node the on the left side of its subtree or the smallest node on the right side of its subtree. In either case, the deleted node is replaced with one of the two most middle nodes in its subtree, keeping the integrity of the tree intact.

Lastly, the print function displays the tree with an in-order traversal (Left, Parent, Right). This creates the effect of output counting upwards as it displays the results. The program uses a stack of nodes to store the values as it travels through the nodes. The results are displayed right as the node is popped off the stack.

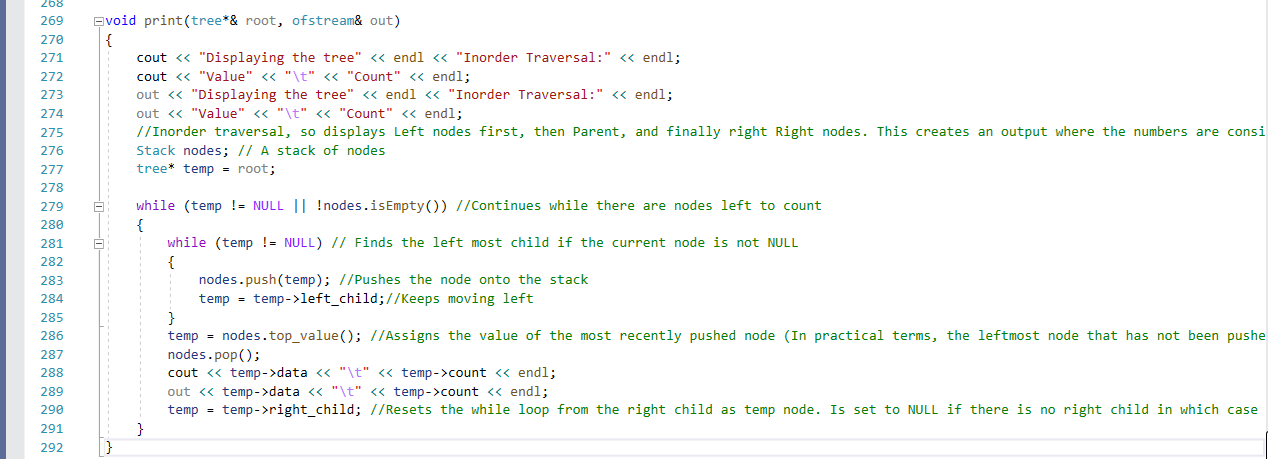


Figure 5: The print function

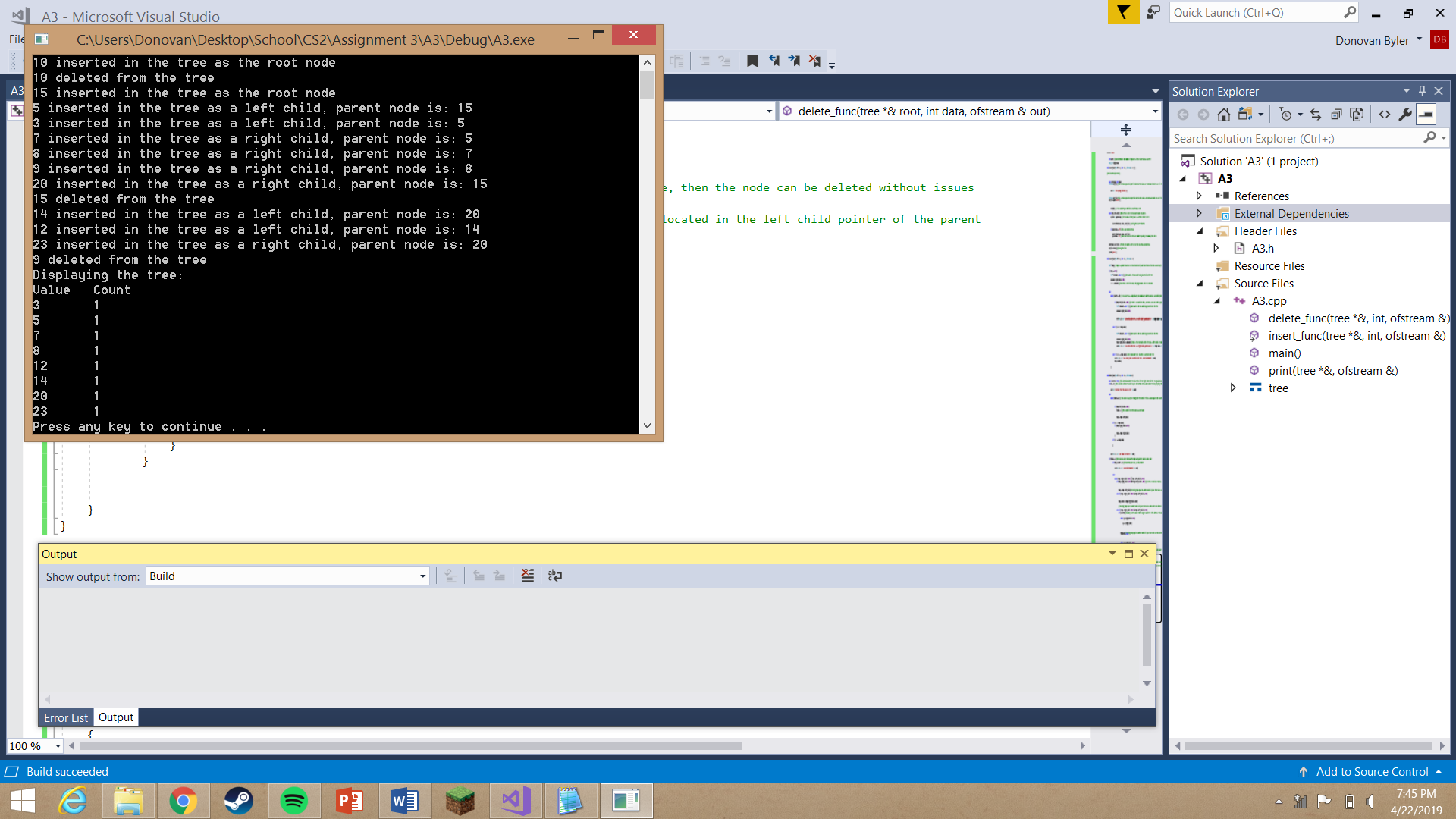


Figure 6: Deletion of the root node