Group Members:
Jessica Cvetkovska - Task 1 and 2 debugging
Dhruv Pandey - Task 1
Mitchell Koski - Task 2 and Task 1 debugging

Performance difference you would expect with vs without balancing. This includes both the insert/remove performance and finding performance - Task 2

	With balancing	Without balancing
Insert/Remove Performance	Keeps the Binary tree with as minimum entries as possible inserting new values into the tree is extremely efficient and costs little on the cpu and memory bus.	There tends to be an extreme difference in the tree causing it to be lopsided. Inserting/removing costs more memory and CPU thus causing an increase of about 2^(n) value of inputs. Significantly slower than using a balanced tree
Finding Performance	Is extremely quick and is done in the most efficient way possible.	Can be quick, however can also be really slow as the amount of entries increases 2^(n)

Failing Reasons:
The way the binary tree is set up is completely incorrect in that the insert function does not properly increase the inputted word. If the word is the exact same, it should increase the inbuilt counter, however it instead throws an duplicate word error. The error should not be happening as it is finding the correct spot in the tree but not increasing the value stored at that part of the tree, hence the error. Also, while the balancing function was working, the print functions were not set up correctly at all, as they should have returned the count of the number of times said word was reached. However, it is returning the memory address, and as such it is not doing what it's supposed to do. This portion of code was clearly not tested before reaching my hands - Mitchell Koski. While the handling of the file and word function do work and can print out, the interaction with the tree is completely messed up and needs redone before resubmitting.