1 Homework Assignment 6

The dictionary search done in Lab Assignment 9 using a `HashSet` shows that a `HashSet` is a very useful structure. In this assignment, you are to implement your own binary search tree for doing lookup; this will be a different method for storing and retrieving than is done by a `HashSet`.

The website provides you with three input data files:

1. the full text of an academic paper;
2. the sorted full text from the paper;
3. the text from the paper sorted in reverse order.

The big difference between the sorted and unsorted versions is that if you implement a binary search tree with data that is already in sorted order, you will get a worst-case binary search tree that is just one long list extending down.

You have several `interface` files that your code must `implement`, and you should notice that this code uses `generics`. your `Record` need have nothing as the data payload other than a `String` that is the data `element`.

Note the sample output. If we define the *depth* of the root to be 0, and then the depth of any node to be 1 larger than the depth of its parent, then every node has a depth value that can be assigned to it. In a perfectly balanced tree, the maximum depth of a tree of N nodes should be $\lg N$ and there should be 2^h nodes of depth h. You are to write code that will dump the tree in *inorder* fashion, which for a binary search tree should be in perfectly sorted order. (It's not sufficient for being correct that you get things in sorted order when you print in inorder fashion, but it is necessary that this be true.) As you traverse the BST in inorder fashion, you are to compute the depth of every node and to print out any new *maximum depth* that you encounter. You must also at the end of this traversal be able to print out the number of nodes and the average depth as well as a frequency count of the number of nodes of any given depth. This last can be done with an array, and you are free to dimension this array fixed at 200 elements. This number should be declared as a `final` constant at the beginning of your program so there will be no magic numbers in the body of your code.

Hooks for a preorder and a postorder traversal method have been left in the code, but you are not required to implement those for this assignment.