Efficiency Write-ups for Program 5
Siqi Wen CS 202

This is a Dictionary Program for terms of Python.

There is a base class called: library, which contains three data members: three String references (term, meaning and example). In this class, there is a method that will prompt and read in information for the term and store them into the data members. There are two compare functions, one takes a String reference as an argument, another takes a node reference as an argument. They both compares the data member (term) with the argument to see if they're match or which one is bigger in value. There is a copy function which will assign the data member to refer to the same contents the argument is referring to. There is also a display function that will display the term, meaning and example for a term.

There is a node_li class derived from the library class. In this class, there are two data members: two references to its own type (left and right). There are functions that will return both the left and right references by value; there are also two other functions that will connect the left and right references with the argument. (This is very important for the recursive functions that we will be using in the tree class.)

And the data structure I used was a binary search tree. The terms are sorted based on the alphabetic value of the term. So there is also a tree class that only contains one data member: a reference to the node_li type. In this class, we have a insert function that will call a recursive function to complete the mission to insert a new term (node) into the tree. These two functions both return a node_li reference by value so that we can connect up the references while unwinding the stack. There are two functions for retrieving a term in the tree. One of them is a recursive function. They both return a boolean value which shows if there is a matching term in the tree or not. There are two functions for removing one term in the tree, one of them is a recursive function. Both of them return a node_li reference by value. In main, we would need to call the retrieve function to check if there's a match, if there's a match, then we would call this remove_one function, otherwise, we would not. Also, for the special case (when the matching node has two children), we need to find the in-order successor to replace the matching term with, then remove the matching term. There are also functions for displaying all terms in the tree, one is a recursive function.