TCSS143
Fundamentals of Object-Oriented Programming-Theory and Application
Programming Assignment 6

The purpose of this programming project is to demonstrate a significant culmination of most constructs learned thus far in the course. This includes Classes, accessors, mutators, constructors, implementation of Comparable, Comparator, use of Collections sort, properly accessing fields of complex objects, fundamental File I/O, **and most of all, a better understanding and use of the LinkedList class and Iterators.**

As such, you are ONLY allowed to use the LinkedList ADT to solve this exercise. **Using any other java ADT to solve this exercise will be penalized a minimum amount of 60%.** To be clear, aside from the standard imports for file I/O, only the following classes/interfaces are allowed: `Arrays, Collections, Comparator, Iterator, LinkedList, List.`

BACKGROUND
Look over Programming Project #4 at the end of the Searching & Sorting Chapter (13), page 891. Programming Assignment 6 is similar with some added features as described below.

You will submit a single file **Programming6.zip** through the Programming Assignment 6 Submission link on Canvas. This zipped file will contain ONLY the 5 files which make up the solution to this assignment. DO NOT zip any folders. You MUST name the file that contains main (the driver) **Assignment6.java**.

The standard grading rules (documentation included) for all previous assignments applies here as well.

DETAILS
To better prepare you for assignments in more advanced courses, specific step-by-step, method headings, class names, and the like have been omitted. What remains is a general description (you may need to read this several times to fully understand the requirements):

You will create a LinkedList of Word objects using all the words found in the input file **words.txt**. A Word object contains 2 String fields; 1 to store a word in its normal form and the other to store a word in its canonical form. The canonical form stores a word with its letters in alphabetical order, e.g. bob would be bbo, cat would be act, program would be agmoprr, and so on. The class Word constructor has the responsibility of storing the normal form of the word in the normal form field and converting the normal form into the canonical form which is stored in the canonical form field (you should call a separate method for this conversion purpose).

Once all the words from the input file have been properly stored in a LinkedList of Word, you should use Collections to sort this list ascending alphabetically based on the canonical words by making the Word class Comparable.

Using an Iterator on the LinkedList of Word, create a 2nd list (new LinkedList) consisting of objects of a new class named AnagramFamily. AnagramFamily should contain at least 2 fields; 1 to hold a list of "Word" words that are all anagrams of each other (these should all be grouped together in the original canonical sorted list), and the 2nd field to store an integer value of how many items are in the current list. (Keep in mind, because the original list contains both the normal and canonical forms, as the AnagramFamily List will also have, a family of anagrams will all have the same canonical form with different normal forms stored in the normalForm field of the Word class). Each AnagramFamily List of Word should be sorted Descending by normal form using a **Comparator** of Word (if you insert Word(s) into a family one at a time, this presents an issue on how to get this list sorted as each Word insertion will require a new sort to be performed to guarantee the list is always sorted. For this reason it is best to form a list, sort it, and then create an AnagramFamily by passing the sorted list to it).

Sort the AnagramFamily LinkedList in descending order based on family size by use of a **Comparator** to be passed to the Collections sort method.

Next, output to a file named “out6.txt” the top five largest families then, all families of length 8, and lastly, the very last family stored in the list.” **Be sure to format the output to be very clear and meaningful**, i.e. Headings (Top 5 Families, Families of Size 8, etc.), Family size, the canonical form for each family, and the normal Word form of the family members displayed horizontally (left to right, **not** vertical one Word per line).

Finally, the first 4 people to complete the assignment should post their output results to the Canvas discussion forum for the remaining students to see the correct answer.

Be sure to instantiate new objects down to the primitive or immutable fields whenever transferring data from one object to another to avoid issues related to shallow copying. Also, be sure to include various methods for manipulation and access of fields as well as helper methods to reduce code in main, such as the input/output of data/information (like all other assignments, you will be graded on decomposition, i.e. main should not contain too many lines of code).

Keep in mind such items as proper documentation (including javadoc), meaningful variable names, proper indentation, reduction of redundancy whenever possible, and so on.

Part of your grade will depend on time. If written correctly (use of iterators and care taken when creating the anagram families), the running time should be less than 3 seconds. Programs that take longer will lose points based on the time. As encouragement to consider all options for speed, programs taking 1 minute will receive a 40 point deduction. Any longer than 3 minutes will receive only minimal points (10) for effort.

Though the basic algorithms involved are straight forward enough, there is a great deal of complexity involved with various levels of access to specific data. As mentioned before and never so importantly as with this assignment, start early and set a goal for completion by this weekend. Trust me, this is sound advice.

As a reminder, you will create at least 5 files: the driver (the driver MUST be named **Assignment6.java)**, Word class, AnagramFamily class, and 2 comparators: 1 to compare Word objects for sorting descending based on the normal form of Word objects and 1 to compare AnagramFamily sizes for a descending sort.

Do **NOT** use any constructs that have not been presented in the course thus far.

You will submit a single file **Programming6.zip** through the Programming Assignment 6 Submission link on Canvas. This zipped file will contain ONLY the 5 files which make up the solution to this assignment. DO NOT zip any folders.