Project 3

Due: May 20 11:59 PM

(but there is nothing wrong with finishing earlier!)

You will be doing an object oriented implementation of project 1 Part A. You will also use data structures such as arrays and vectors freely (unlike then).

This involves the following steps:

Identify ADTs:

Go through the spec and identify all ADTs. You are required to at least have the following ADTs:

- Sentence: Data should include an appropriate data structure to store the sequence of words corresponding to a sentence. Public operations should include sentence input, and the breaking of sentences into words. Don't forget that the details of what a word is are not relevant to this ADT!
- Word: You can decide for yourself what you want here.
- Noun/Verb/Adjective/Pronoun (4 ADTs): Public operations should include boolean functions to check whether an argument Word is a noun/etc.

The above is only a minimum requirement, and you will probably want more ADTs and more functions for these ADTs.

Your design should not require you to access the dictionary files more than once each. File I/O is expensive, and you no longer need to do this since you can read the files once and store it in appropriate data structures.

Object Oriented Design:

Write all the class definitions corresponding to your ADTs. Make sure to follow sound software engineering guidelines, especially in relation to information hiding. This means you should use accessor/mutator functions for everything and not make members public.

There is one additional requirement for the noun/verb/adjective/pronoun ADTs: You must use overloaded >> operators and constructors to initialize them. Since a goal of this project is to learn these, you must show the use of both initialization types (so you can do some of them using >> and others using constructors).

Coding:

Write the code for all your functions! You can borrow from your program 1 code if you want. However, since you now have data structures, some of the code will change.

Testing and Debugging:

You can probably reuse the test data from Project 1. But if you did an inadequate job then, you probably want to rethink your approach. Don't forget - you also know about ddd now to make debugging easier.

Program and Submission Guidelines

As always, you should have the following header on all programs:

```
/*
  Author: <name>
  Course: {135,136}
  Instructor: <name>
  Assignment: <title, e.g., "Lab 1">

  This program does ...
*/
```

As with 136, a program that does not run on the computers in Lab G is considered incorrect. You are graded on the same guidelines as 136. For the testing component of the grade, include a file named "testsentences" containing all sentences you tried.

Submit the program file named <firstname>.<surname>.prog3.cpp. You should also submit your input test file. Although this only counts towards your 135 grade, it will be submitted to your 136 instructor for grading. There will be a submission link on your 136 bb page.

The structure component is a substantial portion of your grade in this program. This means you should have a good object-oriented decomposition of your program. Good decomposition is an art rather than a science, and we will not be grading Davinci better or worse than Michelangelo. However, some decompositions are clearly not in keeping with OO guidelines, and these will not get a good structure grade.