2 Homework Assignment 2

Your assignment for Homework 2 is to improve your Homework 1 code and to run it on some more data to verify that it is a general purpose DLL program.

"First you make it right; then you make it better."

2.1 Basics

The basic assignment is to convert your Homework 1 DLL code to use a generic data payload instead of hard coding the `Record` payload, and then to demonstrate that it works by testing it on the old `Phonebook` code from Homework 1 and on some new data that will be a new "application". You may borrow from all the example code and code fragments in the text. But in essence, you need to convert your code to look like Figures 5.1 through 5.5 so as to remove all hard-coded reference to the `Record`.

Think for a minute before you start about the strategy for changing your code. You have a working `Phonebook` application that uses a working `DLL` implentation for a data structure, and that uses `Record` as the data payload.

First you should change `DLL` to make it use generics, then change `Phonebook` only the minimal amount needed to use the new `DLL`, and then and only then should you swap in a new `VoteSummary` application to replace `Phonebook` and a new data payload `OneLine` to replace `Record`. This is the basic scientific method–change one variable at a time so you know, when things change (i.e., stop working), where to go look for where you have introduced the errors. It's usually just not smart to do radical surgery and *then* try to test.

2.2 Initial Testing

Having made the changes to use generics instead of hard coding the data payload, you should test your new code on the Homework 1 data and verify that you can get exactly the same output. (Well, there may be some minor changes if you decide you want to clean up things, but the "content" of the output should be identical.)

2.3 The Actual Assignment

Now, to verify that your code actually does use generics properly, rework your code to run on the **unsorted** data file of Lab 4. To do this, you will need to change the old `Phonebook` class to be a class called `VoteSummary` and you will need to change the data payload class to be the `OneLine` class from Lab 4. You will also need to code up the inner insertion loop of an insertionsort to be able to link new `OneLine` instances into the appropriate locations in the linked list.

Then read the unsorted data from precinct 305 and insert the data into the linked list so that the linked list is in sorted order.

Note that what you are really doing is this: instead of using an `ArrayList` and an insertionsort inner loop, you are using your own `DLL` and doing an insertionsort on that list.

It is not absolutely required that you create output that is identical to the sorted list that is the output of Lab 4. However, you will learn in CSCE 215 and elsewhere that there is a large set of console commands in a Unix/Linux system that will compare files, count lines, grab lines with specific character strings, etc. Using these tools to show that your "new" correct(?) output is the same as the original correct output can speed up your testing, but that requires you to get the formatting right so the files will be the same down to the exact spacing.