1 Homework Assignment 5

It is now time to count votes in a mature, adult, fashion.

Start with your Lab07 code. That should give you much of what you need to finish out the counting of votes.

Instead of the `ArrayList` with the sorting mechanism, you are in this assignment to use a Java `TreeMap` to collect and keep votes.

The `TreeMap` is a data structure that allows you to use a “key” in order to store a “value”. You should Google “java 6 treemap” to get the official documentation for the `TreeMap` and then read that carefully.

You can comment out your insertion sort code. You won’t need that. You can comment out your references to an `ArrayList`. You won’t need that. You don’t need to worry about reading data into memory, or about reading a file multiple times. Using the `TreeMap` you can read the data file exactly once and let the `TreeMap` structure keep track of the information that is the vote count.

As you read through each `OneLine` you should create a `key` for the `TreeMap<String, Integer>`. The key will consist of the concatenation of the `contest` and `candidate` strings. The `Integer` value stored will be the vote count for that key. Using that concatenation as the key to the `TreeMap`, do a `get` on the `Integer` that is the value for that key. That `Integer` should be the vote count total for that `contest/candidate` key.

Should you pull up a `null` value, that means that this is the first time you have accessed the `TreeMap` for that key. So store a value of 1 for this first vote for that `contest/candidate` key. Every other time you pull up an `Integer` value with that key, you should increment the value (for this new/additional vote for this `contest/candidate`) and then put back the new value of the. `Integer`.

This is the way that you will count, in one pass through the file, without needing to store all the individual vote data.

What you get with a `TreeMap` is the *effect* of random access, with what looks like a single probe, by a key into an array of data (the vote counts). You don’t actually get single-probe access (as with a subscript into an array or `ArrayList`), but the `TreeMap` stores data using a very efficient log-time-search-tree, so it winds up being quite fast as well as being simple to use.