Central Washington University
College of the Sciences
Department of Computer Science

CS-302 Adv. Data Structures & File Proc. Fall 2015

Lab Practice 5

This lab will help you see the advantages of divide and conquer sorts like MergeSort over elementary sorts like insertion Sort. At the same time, we are going to learn some geography!.

Reminder: as stated on the syllabus, it is *very important* that you complete the lab, within two or three days. Tutor time has been set aside to help.
As usual, you find the source files in `/home/cs-302/Labs/lab05` . Remember you cannot write in this directory, so please copy required files to one of your directories.

1. The file `alpha-cities.txt` contains sorted geographic names and population of cities on the west coast of South America. Notice that the order of the countries is north-south.

   ```
   Colombia
   Ecuador
   Peru
   Chile
   ```

   The information in that order

   ```
   Country Provice Name Population
   ```

   Create a class `City` , implementing `Comparable<E>` aimed at ordering the file alphabetically, by city name. Also you may want to override the toString() method to be able to show the contents of the elements easily.

2. Create a client class that reads the file into a into an array. Incorporate a good elementary sorting method (for example `insertionSort( )`). You may copy them directly from one of the programs in `WeeklyPrograms`.

3. Sort the data into a file `wsa-sorted1.txt`. Peruse the file, and see if you have cities with the same name. You may find several cities with the same name. By looking at the corresponding Country, what nice property was present in the sort that you used?

4. In the directory were you find these files, you can also find a merge sort implementation. Incorporate this merge sort (and its private helper methods) into the sort collection (where you had *insertionSort()*, for example), and sort it using merge Sort.

5. Make sure you can compile and run both versions on the server. You may or may have not notice the difference in time, with naturally MergeSort being faster. In any case, you can get numbers for you runs, by using the `time` utility, which provide a very crude but useful form of timing. For example

   ```
   time   java   SortClient
   ```

   The actual command line depends on how you inplemented the reading of the file. Also, it is a good idea to disable output, before timing.