CSCI-1200 Computer Science II — Spring 2006
Homework 6 — Caller ID Maps

In this assignment we revisit a problem from the first test: creating a caller ID system to associate people and their phone numbers. Now that we've learned about the standard library's associative container object (`map`) we can make this system quite efficient and elegant. *Please carefully read the entire assignment before beginning your implementation.*

Your program will expect two command line arguments, an input file and an output file. The input file format is similar to homework 4. Each line begins with a character signaling which of four operations should be performed:

`'a'` This operation will *add* a person and phone number to the phone book. If the person is already in the phone book, a message indicating that their number has changed is output. If the number was in the phone book with someone else's name, an additional message is output indicating that the previous owner must have moved and the previous owner is removed from the phone book.

`'n'` This operation will *lookup the number* for a person. If the person is not in the phone book, an appropriate message is output.

`'i'` This operation will *identify the caller* given a particular number. If the number is not in the phone book, a message indicating an unknown caller is output instead.

`'p'` Finally, this operation is used to *print out the contents* of the phone book: first sorted by name and then sorted by number.

Examples of the messages your program must output are available on the course website. Please follow these examples exactly. To see if your program is performing perfectly, you may use the Unix library program, `diff` which takes two files as arguments and outputs the differences between them. `diff` is included with Cygwin, which may already be installed on your laptop. WinDiff is another option for Windows users. Please see a TA or the instructor in office hours if you have a question about these programs.

The `'a'`, `'n'`, and `'i'` operations should have sub-linear expected running time. The `'p'` operation should have linear expected running time. *Hint: That means you should use maps. In fact, you'll need at least two of them!*

Most of the points for this assignment will be earned through elegant code design and the appropriate use of the `map` container class. For full credit your program should also run the second sample input file which has extra whitespace and inconsistent capitalization, etc. This is a chance to practice various string and character operations. You are not required to create any new classes when completing this assignment, but please do so if it will improve your program design. We also expect you to use `const` and pass by reference/alias as appropriate throughout your assignment.

Do all of your work in a new folder named `hw6` inside of your CSII homeworks directory. Use good coding style when you design and implement your program. Be sure to make up new test cases and don't forget to comment your code! Please use the provided template `README.txt` file for any notes you want the grader to read. **You must do this assignment on your own, as described in "Academic Integrity for Homework" handout. If you did discuss the problem or errors messages, etc. with anyone, please list their names in your `README.txt` file.** When you are finished please zip up your folder exactly as instructed for the previous assignments and submit it through the course webpage.