Cal Poly Spring 2013	**CPE/CSC 365** Introduction to Database Systems	Alexander Dekhtyar Eriq Augustine

Lab 1: Why Databases? Part II

Due date: Wednesday, April 10th, during the lab period.
Note: Lab 2 will be distributed on April 10thas well.

Honor System

This part of the assignment is handed to each group separately, upon submission of the first part of the lab.

While each group will receive the same assignment, I ask you to use the honor system and not directly convey the contents of the assignment to students from the teams that have not yet completed Part I of the assignment. Please let other teams concentrate on finishing the first part, without worrying about the content of the second part.

Lab Assignment

The Task

In the second part of the assignment, you have to change your program to accommodate for the changes in the input data format. You will also be expected to follow a strict output format.

Recall that the original `students.txt` file had the following format:

```
StLastName, StFirstName, Grade, Classroom, Bus, TLastName, TFirstName
```

Generally speaking, if each teacher teachers in exactly one classroom, it is a bit of a waste of space to put teacher's name against the name of every student. Therefore, we now revise the format of the input file. Now, the original `students.txt` is replaced with a pair of files: `list.txt` and `teachers.txt`.

The format of `list.txt` is

```
StLastName, StFirstName, Grade, Classroom, Bus
```

A sample line from `list.txt` is

```
DROP, SHERMAN, 0, 104, 53
```

("Sherman Drop is a kindergarten student assigned to classroom 104 who takes bus number 53.")

The format of `teachers.txt` is

```
TLastName, TFirstName, Classroom
```

A sample line from `teachers.txt` is

```
NIBLER,  JERLENE, 104
```

("Jerlene Nibler teaches in classroom 104.")

The data files are available on-line. The direct URLs for the files (they won't be linked to the web page until the end of lab on April 5th) are:
`http://users.csc.calpoly.edu/~eaugusti/365-files/misc/list.txt`
`http://users.csc.calpoly.edu/~eaugusti/365-files/misc/teachers.txt`

The information in these files is exactly the same as in the original `students.txt` file, so answers to test searches in both versions of your program should be the same.

Your goal is to change the `schoolsearch` program you have developed for Part I of this assignment to handle the new input data formats. In addition, you are asked to design, and implement in the new program extra search facilities.

Specs

With the exception of the new functionality, described below, the new `schoolsearch` program shall have the same functionality as the `schoolsearch` program, working exactly the same way, i.e., it shall accept the same search commands and produce the same results.

The key difference is that now, instead of obtaining the data from a single file called `students.txt`, your program will read data from two files: `list.txt` and `teachers.txt`. Same assumptions about the location of the files (current directory) and error handling (minimal) apply.

Additional Functionality

You are asked to design and implement additional search functionality. In particular, the current language of search instructions for the program allows one to search for information about a student, given student's last name; find the bus route of a student, given student's last name, list all students attending a class taught by a specified teacher; list all students in a specific classroom; and list all students taking a specific bus route.

Three new searches need to be added to the program:

- Given a grade number, list all students assigned to it. (`G[rade]: <Number>`).
- Given a classroom number, find the teacher (or teachers) teaching in it[1]. (`C[lassroom]: <Number> T[eacher]`).

[1] In our test file, there in one teacher per classroom. However, your search algorithm cannot assume that it will always be the case.

- Given a grade, find all teachers who teach it. (`G[rade]: <Number> T[eacher]`).

You shall

- extend the language of search instructions to allow for these three searches to be specified by users;
- implement the functionality supporting each search.

Output Format

Your program will be expected to follow the exact output formatting described below:

- Each record should be listed one to a line with a comma and a space separating each attribute (first name, last name, classroom, etc.).
- Each attribute should maintain the case that was used in the input file (if it goes in as caps, it should come out as caps).
- The time that each operation took (in ms) should be printed on the line following the last of the results.
- The response to each query should have no additional output besides the results and the time for the search.
- When the user quits, your program should say 'bye!' to the user followed by a newline.

Example output (user input is **bold**):

```
C: 104
WOOLERY, NOLAN
VILARDO, EMMANUEL
13ms
Student: SCHOENECKER
SCHOENECKER, PHUONG, 6, 109, GAMBREL, JAE
7ms
Quit
bye!
```

Implementation notes

Obviously, there are two ways to do this task. One way is to start from scratch and build a brand new program. The other way is to adapt the first program to the new data format. The decision of what to do is left up to individual teams.

Testing and Deliverables

You have to submit your code, a `README` file describing the syntax of the new search commands and instructions on how to compile (if needed) and run your program, and a write-up.

The write-up shall contain the same information as the write-up for the first part of the assignment. In addition, I encourage each member of each team, write some additional comments about the assignment: What do you think is the purpose of the assignment? What have you learned?

Submission. Use `handin` as follows:
$ handin eaugusti lab01b <FILES>

Good Luck!