Lab 9: Welcome to CSC 365!

Due date: Friday, December 4, 11:59pm

Lab Assignment

Assignment Preparation

Lab type. This is a **team programming lab** done in teams of two students. The team assignments are done by the instructor and are announced when the assignment is handed out. Note that because this is a **team** lab, there is no expectation that you and your partner will sit next to each other to develop the entire system, altough pair programming approach is still encouraged.

Purpose. The lab allows you to practice the use of structures, strings and file I/O in your programs.

Programming Style. All submitted C programs must adhere to the programming style described in detail at

http://users.csc.calpoly.edu/∼cstaley/General/CStyle.htm

When graded, the programs will be checked for style. Any stylistic violations are subject to a 10% penalty. Significant stylistic violations, epsecially those that make grading harder, may yield stricter penalties. Also note the the Lab 2 requirement for the content of the header comment in each file you submit applies **to each assignment** (lab, programming assignment, homework) in this course.

The Task

A similar assignment is handed to students in CSC/CPE 365, Introduction to Database Systems as their Lab 1. The purpose of this assignment is to build a simple text-based database application, complete with a simple query to use.

You will write an application that allows the uses to browse information about students and teachers of a local elementary school. This information is originally stored in two text files: `list.txt` and `teachers.txt`. Each file stores a list of records. The records in `list.txt` store information about students; the records in `teachers.txt` — information about teachers. Each records is stored in a single line of the file. The records consists of fields and the field values in the files are **space-separated**.

The format of `list.txt` is

```
StLastName StFirstName Grade Classroom
```

Here, `StLastName` and `StFirstName` are the last name and the first name of a student, `Grade` specifies the grade the student goes to and `Classroom` specifies the classroom where the student studies. `Grade` and `Classroom` are integers (Kindergarden is 0), while all other fields are strings.

A sample line from `list.txt` is

```
DROP SHERMAN 0 104
```

The line can be interpreted as *"Sherman Drop is a kindergarden student assigned to classroom 104."*

The format of `teachers.txt` is

```
TLastName TFirstName Classroom
```

Here, `TLastName` and `TFirstName` are the first and last name of the teacher, and `Classroom` is the classroom the teacher teaches in.

A sample line from `teachers.txt` is

```
NIBLER  JERLENE 104
```

It is interpreted as *"Jerlene Nibler teaches in classroom 104."*

Your goal is to write a program, which searches the `list.txt` and `teachers.txt` files and outputs the results of the search. The following searches have to be implemented:

- Given a student's last name, find the student's grade, classroom and teacher (if there is more than one student with the same last name, find this information for all students);
- Given a teacher, find the list of students in his/her class;
- Find all teachers who teach at a specified grade level;
- Find all students at a specified grade level;

Formal Specs

You are two write a C program schoolsearch.c, which implements the requested functionality. The program must satisfy the following requirements.

R0. Your program shall declare two structure types student and teacher to represent a single record from the list.txt and teachers.txt files respectively. It shall, then declare two arrays: one containing the student records and one containing the teacher records. Upon start, your program shall read the contents of the files list.txt and teachers.txt into the respective arrays. All other processing will be conducted with the data stored in these two arrays.

Use the following restrictions:

Maximal number of students:	200
Maximal number of teachers:	50
Maximal length of a first/last name:	30
Range of grades:	0 — 6
Range of rooms:	100 — 699

R1. After the program reads the information from the list.txt and teachers.txt, it shall provide the user with a prompt. Following that, the program will, in a cycle, read search instructions entered by the user, conduct the appropriate search, print the result, and wait for the next user instruction until the termination command is recieved.

R2. Your program shall implement the following language of search instructions:

- S <lastname>
- T <lastname>
- G <number>
- GT <number>
- Q

Notes: For simplicity all instructions are case sensitive. In the specs above, items in angle brackets (<>) are the values provided by the user.

R3. S <lastname>

When this instruction is issued, your program shall perform the following:

- search the list of students for the entry (or entries) for students with the given last name.
- For each entry found, print the last name, first name, grade and classroom assignment for each student found.
- Search the list of teachers for the teacher assigned to the student's classroom. Print the name of the teacher (last and first name).

Only one line of output per student shall be produced. If no students with given last name is found, the program shall output nothing and simply produce a new prompt and wait for the next command.

R4. `T <lastname>`

When this instruction is issued, your program shall perform the following:

- Find all students taught by the teachers whose last name has been entered in the query command.
- For each entry found, print the last and the first name of the student.

Note that student records *do not contain* the last name of their teacher. Instead, they contain the classroom for the student. Teacher records **also include classroom numbers.** So, to find out which students are in the given teacher's class, you have to do the following:

1. Search the list of teachers until you find the teacher with the given last name. Find the classroom for this teacher.
2. Search the list of students for **all** records that have the same classroom number. Report the names of each student found.

R5. `G <Number>`

When this instruction is issued in the form above, your program shall perform the following actions:

- Search the list of students for the entries where the student's grade matches the number provided in the instruction.
- For each entry, output the name (last and first) of the student.

R6. `GT <Number>`

When this instruction is issued, your program shall

- Output the names (last and first) of all teachers who teach the given grade. Each teacher's name shall be printed exactly once.

Note, again, that a teacher's record does not contain information about a grade (s)he is teaching. But you have information about the classroom for each teacher. On the other hand, student records contain information about both the classroom and the grade of the student.

To correctly process this command, we need to make one important assumption:

All students in each classroom attend the same grade.

(A condition like this is called in *database theory* a functional dependency.)

To correctly process this query, you need to do the following:

1. Scan the list of students for records which match the input grade.

2. For each record found, find the classroom.

3. Scan the list of teachers for a teacher who teaches in the given classroom. Output their full name (last and first).

4. Note, that you need to make sure to look for each classroom number **no more than once**, otherwise you will be outputting the name of the same teacher multiple times. There is a number of ways to ensure that. A simple (although not the most efficient method) is to create an array of truth values - one for each teacher, and once the teacher's name is output, set the appropriate element of the array to true. Then, each time you want to output a teacher's name, just check whether the appropriate truth value is set to false (and then print the name) or true (do not print the name).

   Other methods can be used as well. I am happy to discuss options with each team.

R7. Quit. The command

```
Q
```

is the quit command. If this command is entered, your program shall stop its work.

R8. The program shall assume that the files `list.txt` and `teachers.txt` are located in the directory from which it is run.

E1. Your program can have minimal error-checking. If either `list.txt` or `teachers.txt` is not available, or has the wrong format - exit the program. If the `list.txt` or `teachers.txt` file has more records than the maximum allowed number, quit the program. If the search instruction has different syntax than what is prescribed in **R2**, go back to the prompt.

Implementation Notes

main(). The `main()` function of your program shall contain the declarations for the main data structures used in the program: the arrays for the lists of students and teachers. It also shall contain the main loop of the program: output the prompt, read input, recognize which command needs to be executed, execute it, and output results, if any.

Query processing functions. Each of the four types of queries/commands shall be implemented as a separate function. The following function declarations can be used:

```
/* for  S <Name>   command */
void getStudentInfo(<StudentType> studentList[], <teacherType> teacherList[],
                    char name[], int nStudents, int nTeachers);

/* for  T <Name>   command */
void findStudents(<StudentType> studentList[], <teacherType> teacherList[],
                   char name[], int nStudents, int nTeachers);

/* for  G <Number>   command */
void getGradeList(<StudentType> studentList[], int grade, int nStudents);

/* for  GT <Number>   command */
void findGradeTeachers(<StudentType> studentList[], <teacherType> teacherList[],
                       int grade, int nStudents, int nTeachers);
```

Here `<StudentType>` and `<teacherType>` are the names of structure types you declared for the student and teach records respectively; `studentList` and `teacherList` are the arrays of student and teacher records respectively; `name` and `grade` are the inputs from the respective command (either a student/teacher name, or a grade number) and `nStudents` and `nTeachers` are respectively the number of students and teachers read from the input files.

The functions return no values, but **are expected to print the answers to the queries**.

Other functions. You may find that other functions may be useful for various parts of the program. For example, you can declare `int checkTeacherName(...)` function (with appropriate to your solution input) to check if you have already printed the name of a teacher in repsonse to a `GT <Number>` command.

Feel free to consult me about specific functions that you may want to declare. There is no requirement that you declare/define any functions other than the four functions processing individual query types.

Submission.

Files to submit. Each pair submits one set of files from one account. The following files are mandatory:

> `team.txt`, `schoolsearch.c`,

Additionally, if you have developed other files (extra .h file, for example), submit them as well.

team.txt file shall contain the name of the team and the names of all team members in each pair, and the Cal Poly IDs of each. E.g, if I were on the team with Dr. John Bellardo, my team.txt file would be

```
Go, Poly!
John Bellardo, bellardo
Alex Dekhtyar, dekhtyar
```

Submit team.txt as soon as you form your group.

Files can be submitted one-by-one, or all-at-once.

Submission procedure. You will be using handin program to submit your work. The procedure is as follows:

- ssh to vogon (vogon.csc.calpoly.edu).

```
> handin dekhtyar lab08  <your files go here>
```

handin is set to stop accepting submissions 24 hours after the due time.

Testing

The list.txt and teachers.txt files are available to you from the Lab 9 "tests and data" page. We will be using **only** these files to test your program.

Sample Run

```
$ schoolsearch
Elementary School Database

-->G 0
NOGODA, ISMAEL
PREHM, SHANEL
GELL, TAMI
DROP, SHERMAN
BUSTILLOS, HILMA
HOUTCHENS, THEO
BRINE, FRANKLYN
BYRUM, BENNIE
NETZEL, JODY
VANVLIET, COLLIN
HONES, GUILLERMINA
GRABILL, JULIENNE
MOWATT, KITTIE
NAKAHARA, SHERON
```

```
SOLOMAN, BRODERICK
WIRTZFELD, DELORAS
-->S NOGODA
NOGODA, ISMAEL, 0, 105, MARROTTE, KIRK
-->T MARROTTE
NOGODA, ISMAEL
BYRUM, BENNIE
NETZEL, JODY
MOWATT, KITTIE
NAKAHARA, SHERON
-->GT 0
NIBLER, JERLENE
MARROTTE, KIRK
TARRING, LEIA
-->Q
$
```