CS 240: Programming in C
Fall 2020
Homework 3

Prof. Turkstra

Due September 23, 2020 9:00 PM

1 Goals

In this assignment you will read a data file into arrays and manipulate the data in the relational arrays.

2 Getting Started

1. Log in to one of the Purdue CS Linux systems. These include data.cs.purdue.edu and borgNN.cs.purdue.edu where NN=01,02, etc
2. Inside the cs240 directory, setup a hw3 directory by running the following commands...

   ```
   $ cd cs240
   $ git clone ~cs240/repos/$USER/hw3
   $ cd hw3
   ```

3. This creates your hw3 directory, and also copies what is called a **Makefile** into it. This Makefile has a number of targets defined to make your life easier...

   When you are ready to try your work, you can run:

   ```
   $ make hw3_main
   ```

   to build hw3_main, or run:

   ```
   $ make hw3_test
   ```

   to build the test module. Or, to build both, just run:

   ```
   $ make
   ```

   For this assignment, the test module will produce a number of files in `test_data_files` folder under your directory. Running it many times will result in many files. You can run:

   ```
   $ make clean
   ```

   to clear out these temporary files.

   Note: any time you run make, your changes are automatically pushed to a git repository in the cs240 course account. As mentioned before, we **strongly** encourage you to always work on a Purdue CS Linux system. This mechanism cannot work otherwise.

3 The Big Idea

You will implement a simple relational database to analyze the student enrollment data of Purdue University. We will have a table of college names, a table of genders, a table of student residency and another table of student accommodation types. We will use the index of an entry in one table to access the related information in the other tables. For example, the first college name in Table 1 corresponds to the data in the first row in Table 2 (2106, 1475, 231), the first row in Table 3 (512, 792, 2277) and also the first row in Table 4 (1702, 567, 689, 623).

4 The Assignment

Tables 1, 2, 3 and 4 are the logical representation of the relational tables.

Table 1: College Names

Name
Agriculture
Education
Engineering
Science

Table 2: Genders

Number of Female Students	Number of Male Students	Number of Unlisted Students
2106	1475	231
1131	278	151
3250	9791	320
2144	3787	132

Table 3: Student Residency

Number of International Students	Number of Non-Resident Students	Number of Resident Students
512	792	2508
116	373	1071
3744	6160	3457
1766	2081	2216

Table 4: Student Accommodation Types

Resident Hall	Rent House	Rent Apartment	With Parents
1702	567	689	854
1000	203	159	198
6134	3021	2998	1208
2092	1872	794	1305

The first table will be a two-dimensional character array (which can be viewed as a one-dimensional string array).

The second table will be a two-dimensional array of `ints`. The first index will correspond to a particular college (row), the second index to the particular column, numbered starting at 0. For instance, index 0 contains the number of female students.

The third table will be a two-dimensional array of `int`s, similar to the second table.

The forth table will also be a two-dimensional array of `int`s, similar to the second and third table.

You will design a statistical analysis program that will:

- Read the data file into the tables (arrays);
- Determine the total number of students of a certain college;
- Determine the percent of students living with parents of a certain college;
- Determine the total ratio of students living in a resident hall to students renting a house or an apartment of a certain college;
- Determine the total ratio of female students to male students at Purdue University;
- Determine the college with the greatest number of students;
- Determine the college with the greatest percent of international students;
- Output data according to the input data file.

Examine hw3.h and the input file specifications that follow for more information. The necessary information on opening/reading from/writing to files is both in the course text and lecture slides.

4.1 Functions You Will Write

`int input_tables(char *in_file);`

This function will open the file specified by the first argument and read in the records, filling the `g_college_name[]`, `g_student_gender[]`, `g_student_residency[]` and `g_student_accommodation[]` arrays. This function should return the total number of records (colleges) read. The function should also set `g_college_count` to be the number of colleges read in from the file (even if an error occurred). If any errors occur, you should instead return an appropriate error code as described in Section 4.4.

`int student_size(char *college_name);`

This function takes the college name as an argument and outputs the total number of students in the given college.

`float live_with_parents_ratio(char *college_name);`

This function takes the college name as an argument and outputs the percent of students living with parents of a certain college.

`float on_off_campus_ratio(char *college_name);`

This function takes the college name as an argument and outputs the total ratio of students living in a resident hall to students renting a house or an apartment of a certain college.

`float gender_ratio(void);`

This function should compute and return the total ratio of female students to male students at Purdue University.

`int max_student_size(void);`

This function should compute and return the index corresponding to the college with the largest number of students. If there exists more than two colleges that have the same largest number of students, then this function should return the smallest index among these colleges.

`int max_international_percent(void);`

This function should compute and return the index corresponding to the college with the largest percentage of international students. If there exists more than two colleges that have the same largest percentage of international students, then this function should return the smallest index among these colleges.

`int output_tables(char *out_file, int start, int end);`

This function accepts a character string that represents a filename to open for writing and two integers representing the start and end rows from Table 1. It will return `OK` if no errors occur and output to the specified file the college name, the total number of students in this college, the number of international students in this college, the number of students living in a resident hall in this college and the number of students living with parents in this college for each row from `start` to `end`.

For example, after reading the record below into the arrays with the `input_tables()` function (with all other records), this function, when called as `output_tables(‘‘out.txt’’, 1, 2)`, the result would be:

```
Education;1560;116;1000;47
Engineering;13361;3744;6134;888
```

If any errors occur, you should instead return an appropriate error code as described in Section 4.4.

4.2 Input Files

The input file will contain records of the following format:

```
college_name;female;male;unlisted;international;non_resident;resident;hall;house;
apartment;with_parents
```

- `college_name` (`char buffer[MAX_COLLEGE_LEN]`) - The first field of a record is a character array of max length `MAX_COLLEGE_LEN` that holds the college name.
- `female` (`int`) - Number of female students;
- `male` (`int`) - Number of male students;
- `unlisted` (`int`) - Number of unlisted students;
- `international` (`int`) - Number of international students;
- `non_resident` (`int`) - Number of non-resident students;
- `resident` (`int`) - Number of resident students;
- `hall` (`int`) - Number of students living in a resident hall;
- `house` (`int`) - Number of students renting a house;
- `apartment` (`int`) - Number of students renting an apartment;
- `with_parents` (`int`) - Number of students living with parents.

An example file looks like this:

```
Agriculture;2106;1475;231;512;792;2277;1702;567;689;623
Education;1131;278;151;116;373;920;1000;203;159;47
Engineering;3250;9791;320;3744;6160;3137;6134;3021;2998;888
Science;2144;3787;132;1766;2081;2084;2092;1872;794;1173
```

We have provided a sample input file named "enrollment.txt" in ~`cs240/public/homework/hw3`.

4.3 Header Files

We provide a header file, "hw3.h", for you. It contains prototypes for each of the functions that you will write as well as #definitions for the constants. You should not alter this file. We will replace it with the original when grading.

4.4 Error Codes

- `OK`: no errors encountered;
- `FILE_READ_ERROR`: indicates that the file cannot be opened for reading;
- `FILE_WRITE_ERROR`: indicates that the file cannot be opened for writing;
- `RECORD_ERROR`: indicates unexpected characters/fields/wrong data type in a record;
- `OUT_OF_BOUNDS`: indicates that the entry is not within the expected range;
- `NO_SUCH_COLLEGE`: indicates that the college does not exist in the record;
- `NO_DATA_POINT`: indicates that the input file itself has no data or outputs with no data.

Hints

- Make sure that you create the definitions for `g_college_name`, `g_student_gender`, `g_student_residency` and `g_student_accommodation` in your hw3.c file.
- Make sure that you scan the input lines with a format string that looks like "`%[^;];%d;%d;%d;%d;%d;%d;%d;%d;%d;%d\n`"

These Standard Rules Apply

- You may add any `#include`s you need to the top of your hw3.c file;
- You may not create any global variables other than those that are provided for you. Creation of additional global variables will impact your style grade;
- You should check for any failures and return an appropriate value;
- You should not assume any maximum size for the input files. The test program will generate files of arbitrary size;
- You should not assume that `MAX_COLLEGE_LEN` and `MAX_RECORDS` cannot change;
- Do not look at anyone else's source code. Do not work with any other students.

Special Rules

Just to make sure you know how to use `fopen()` and `fscanf()`, we won't allow you to use `access()`, `feof()`, or `ferror()` for this assignment. In other words, you should just use `fopen()`, `fscanf()`, and `fprintf()`.

Submission

To submit your program for grading, type:

```
$ make submit
```

In your hw3 directory. You can do this as often as you wish. We encourage you to submit your code as often as possible. Only your final submission will be graded.

5 Grading

The operation of your functions will be graded out of 100 points. The point breakdown will be determined by the test program.

The test program will be run many times when grading. It is your job to do the same. The lowest score will be your final grade.

This homework will also have a style grade based on 20 points, with 2 points deducted for each code standard violation found.

Your code must compile successfully using `-Wall -Werror -std=c99` to receive any credit. Code that does not compile will be assigned an automatic score of 0.

If your program crashes (e.g., segmentation fault) at any point during the testing process, your score will be reduced by 25 points.