

# Programming Assignment 6

---

**Due** Dec 2 by 10pm      **Points** 90

---

## Overview

There are multiple goals for this assignment. One goal, of course, is to further develop your C programming skills by implementing a larger program. Another goal relates to the data being processed. For this assignment you will write a program to process real county demographics data collected by the **CORGIS project** (<https://corgis-edu.github.io/corgis/>) from Census information. The most recent version of this data set lacks one of the more interesting pieces of data for this analysis, so we will be using a previous version of the data set. You can find information about it through the **Internet Archive - county demographics data** ([https://web.archive.org/web/20210117162214/https://corgis-edu.github.io/corgis/python/county\\_demographics/](https://web.archive.org/web/20210117162214/https://corgis-edu.github.io/corgis/python/county_demographics/)).

**Warning:** It is common to write programs to analyze data of this sort and to then present "conclusions" as artifacts of the perceived "truth" of the program. You are asked to write a program to analyze this data, but take care to not interpret the results without consideration for what is not presented. In particular, the data presents various metrics for each county (such as the percentage of those within a county with a Bachelor's degree or higher, those in the county of different ethnicities, median income, etc.), but the data *does not* include any causal information that suggests the reasons for these numbers so one must avoid implicit biases in interpreting this data.

A final goal, then, is to carefully reflect on the data presented as output from your program, on the shortcomings of such presentations, and on the potential misinterpretations of such data.

## Course Learning Objectives

This assignment addresses the following course learning objectives.

### Direct


- Read and write complex programs in the C programming language.
- Transition to upper-division coursework where programming is a tool used in the exploration of course-specific content.
- Transition from following stepwise, prescribed decompositions to independent problem solving.

### Indirect

- Use standard UNIX user-level commands and the UNIX software development environment.
- (DLO) Understand the history of issues related to diversity, social and economic inequities, and political power in the United States and across the world.

## Setup

Clone the repository.

- Accept the **GitHub Classroom Assignment 6 invitation**  (<https://classroom.github.com/a/Q9EJvFF5>).
- Clone the repository and examine the provided files. This repository includes a significantly shortened version (`small.csv`) of the full demographics data (`county_demographics.csv`) for use in initial testing, but you will also want to verify that your program works with the full data set.

## Tips

- Give careful consideration to how you will store the information for each county. Certainly a structure is expected, but will you store the data as separate fields? Or maybe parts as an array (of fixed size) of fields according to grouping (e.g., corresponding to the Ethnicities category in the data set).
- Consider using an array of valid field strings (e.g., `"Education.High School or Higher"`) to compare against instead of using a long sequence of `if` statements.
- Use a debugger to inspect your data or print the data to verify that your program has stored what you intended it to.
- Program incrementally. Verify that each component works before integrating them (e.g., if the file is not read properly, then the rest of the program will not behave as expected).

## The Tasks

There are, in effect, two separate tasks listed below. The first presents the details of the program that you are expected to write, whereas the second requires that you provide example inputs to that program.

In addition to the tasks listed here, there is one other task that appears as a separate assignment within Canvas (and that is linked under Deliverables below). This **Data Reflection** (<https://canvas.calpoly.edu/courses/83933/assignments/562938>) will be completed after you finish the program.

# Task 1

Your program must take two command-line arguments. The first argument is expected to be the name of a file containing the county demographics data (as defined by the CORGIS project; more details below). The second argument is expected to be the name of a file containing a sequence of operations for your program to execute on the data (more details below). Your program must validate that these arguments exist (both on the command-line and as accessible files).

## Demographics File

Your program must load the data from the specified demographics file (note that the first line in this file contains "column headers"; these can be skipped). This file is a CSV (comma-separated values) file, which allows for a simpler approach to parsing (see the earlier Tips). Each "field" (as delimited by commas) also includes double-quote characters - we do not want these; in fact, the numeric fields should also be converted to the proper data types (as listed on the CORGIS page).


The required functionality for your program will not use all of the data from this file; you must decide if you want to store this extraneous data or not. Your program must support (a limited set of) queries against the following fields (as denoted by the "column headers"); when only the prefix is given (e.g., "Education"), then all fields with that prefix must be supported by your program.

- County - a string
- State - a string
- Education (all) - floats
- Ethnicities (all) - floats (note that *Ethnicities.White Alone not Hispanic or Latino* has been modified to remove the internal comma from the this field name)
- Income (all) - int (*Median Household Income* contains a typo in the original data set but this has been corrected in the provided files), int (Per Capita Income), float (Persons Below Poverty Level)
- Population.2014 Population - int

## Processing Demographics File

After the demographics file is loaded, print to `stdout` an indication of the number of entries loaded. If an entry in the file is malformed (e.g., there are missing fields or the data within a field cannot be properly converted), then print an error message to `stderr` indicating the line number of the entry, skip that entry, and continue processing.

## Operations File

The operations file dictates the behavior of an execution of your program. There are (essentially) four types of operations, with some variation for a few of these operations. Though this is rather limiting, it is a reasonable start and allows us to avoid effectively implementing a programming language (e.g., **SQL**  <https://en.wikipedia.org/wiki/SQL>). Your program will perform each operation in order as read from the operations file; as such, operations that appear later in the file will use the data available at that point (i.e., after the earlier operations have been performed).

## Supported Operations

Each of the supported operations is defined next, with an example run shown after. Implement and test these incrementally (in whichever order you wish).

When an operation requires a **<field>**, that name is expected to match one of the CORGIS dataset labels (e.g., "Income.Persons Below Poverty Level").

- **display** - Print (to `stdout`) the county information for each entry. There is no required format, but only the information for the required fields should be printed. Do your best to make the output somewhat user-friendly.
- **filter-state:<state abbreviation>** - Reduce the collection of entries to those with matching **state** abbreviation. If no state matches the abbreviation, then the resulting collection will be empty. Print "Filter: state == **<state abbreviation>** (**xyz** entries)" (where **xyz** is the number of remaining entries) to `stdout`.
- **filter:<field>:<ge/le>:<number>** - Reduce the collection of entries to those for which the value in the specified **<field>** is either greater-than-or-equal-to (if **ge** is given) or less-than-or-equal-to (if **le** is given) the specified **<number>** (which is expected to be a float). The only supported comparisons are **ge** and **le**. This operation is not meaningful for all fields (e.g., County), so your program should verify that the data within the specified field is numeric (and that the field exists). Print "Filter: **<field>** **<ge/le>** **<number>** (**xyz** entries)" (where **ge** or **le** is as specified by the operation and **xyz** is the number of remaining entries in the collection) to `stdout`.
- **population-total** - Print (to `stdout`) the total 2014 population across all entries (not per entry) as "2014 population: **xyz**" (where **xyz** is the population total).
- **population:<field>** - Compute the total 2014 *sub-population* across all entries (not per entry). The specified **<field>** is expected to be a percentage of the population for each entry, so this computation requires computing the sub-population by entry. The only viable fields are those with an *Education* or *Ethnicities* prefix and *Income.Persons Below Poverty Level*. Print the result of this computation (to `stdout`) as "2014 **<field>** population: **xyz**" (where **xyz** is the computed total).
- **percent:<field>** - Print the percentage of the total population within the sub-population specified by **<field>** as "2014 **<field>** percentage: **xyz**" (where **xyz** is this computed percentage). This operation is based on the previous two (with the same expectations on **<field>**).

## Processing Operations File

Again, your program must perform each operation in order as read from the operations file; as such, operations that appear later in the file will use the data available at that point (i.e., after the earlier operations have been performed). A valid line within the operations file is either blank or contains a single, well-formed operation. If a line in the file is malformed (e.g., there are missing fields or the data within a field cannot be properly converted), then print an error message to `stderr` indicating the line number of the entry, skip that line, and continue processing.

## Sample Runs

Sample executions with different operations files can be found on the [Assignment 6 Sample Runs page \(https://canvas.calpoly.edu/courses/83933/pages/assignment-6-sample-runs\)](https://canvas.calpoly.edu/courses/83933/pages/assignment-6-sample-runs).

## Task 2

Submit operations files that can be used to determine each of the following.

- Percentage of the population below the poverty level in the counties for which the high school or above completion percentage is  $\leq 80$  percent.
- Percentage of the population below the poverty level in the counties for which the bachelor's or above completion percentage is  $\geq 40$  percent.
- (For an *Ethnicities* column of your choosing) Percentage of the population below the poverty level in the counties for which the selected ethnicity is  $\geq 40$  percent.
- (For the same *Ethnicities* column) Percentage of the population below the poverty level in the counties for which the selected ethnicity is  $\leq 40$  percent.

## Deliverables

- Source Code (and Makefile) and sample operations files - Push all relevant source code and an appropriate Makefile to your repository.
- **Data Reflection** (<https://canvas.calpoly.edu/courses/83933/assignments/562938>)- In the Text Entry box, address the Data Reflection prompt.

<b>Programming Assignment 1</b>			
<b>Criteria</b>	<b>Ratings</b>		<b>Pts</b>
Functionality	<b>60 pts</b> <b>Full Marks</b>	<b>0 pts</b> <b>No Marks</b>	60 pts
Code Review	<b>10 pts</b> <b>Full Marks</b>	<b>0 pts</b> <b>No Marks</b>	10 pts
Valgrind Check Clean valgrind report on the department servers.	<b>15 pts</b> <b>Full Marks</b>	<b>0 pts</b> <b>No Marks</b>	15 pts
Sample Operations Files (Task 2)	<b>5 pts</b> <b>Full Marks</b>	<b>0 pts</b> <b>No Marks</b>	5 pts
			<b>Total Points: 90</b>