Assignment 1

Due: Sunday, 15 April 2018, 11:59 PM on TEACH as a .tar

Structs, File I/O and Review

(90 pts) Implementation

You will write a program to compare state and county information. You must have the following structs in your program.

```
struct county {
  string name;  //name of county
  string *city;  // array of city names in county
  int cities;  //number of cities in county
  int population;  //total population of county
  float avg_income;  //avg household income
  float avg_house;  //avg household price
};
struct state {
  string name;  //name of state
  struct county *c;  //array of counties
  int counties;  //number of counties in state
  int population;  //total population of state
 };
```

You will receive the number of states and filename from the user as command-line arguments. The number supplied with the –s option is the number of states to be created and the text following the –f option is the filename with the state/county information:

a.out –s 2 –f states1.txt

These option and value pairs can come in any order with the option always followed by the value.

The shown example would create a dynamic array of two states on the heap, and you would read the rest of the information about the state and counties from a file. Each line in the file will contain the information for each state and county in the following order:

State_name state_pop #_county

county_name county_pop county_income county_house #_cities city_name

Example:

Oregon 1000000 2

Benton 53000 100000 250000 1 Corvallis

Lane 80000 50000 150000 2 Eugene Springfield

South_Carolina 1000000 2

Anderson 80000 100000 80000 2 Anderson Pendleton

Pickens 50000 50000 20000 2 Clemson Pickens

Your program must define the following functions, with the **exact prototypes**:

```
bool is_valid_arguments(char *[], int);
state * create_states(int);
void get_state_data(state *, int, ifstream &);
county * create_counties(int);
void get_county_data(county *, int, ifstream &);
void delete_info(state **, int);
```

In addition to these functions above, you need to determine the other functions you will need to print information answering the following prompts:

- the state with the largest population,
- the county with the largest population,
- the counties with an income above a specific amount, (You must get input from the user for this)
- the average household cost for all counties in each state,
- the states in sorted order by name,
- the states in sorted order by population,
- the counties within states sorted by population,
- the counties within states sorted by name.

The user should be given the choice to print this information to the screen or a file. If the user chooses a file, then you need to prompt the user for the filename. You can decide how you want to store the information in the file.

You need to separate your files into interface and implementation and create a **Makefile** to handle the compilation. Create a **state_facts.h**, which has the struct type for states and counties, as well as the function declarations for your program. The corresponding function definitions should appear in a **state_facts.cpp** file and your main function in a **run_facts.cpp** file. You

should have a makefile that contains a command to create a state_facts executable and a command to clean your files.

Your program must be able to:

- Print a usage message to the user when too few arguments are supplied or when the options are not –s or -f. You do not need to recover from this, just handle by printing a message and cleanly exit the program.
- Print an error message and recover, when the user doesn't supply a positive, non-zero integer for the states value.
- Print an error message and recover, when the user doesn't supply a valid filename to open.
- Provide the stats for the states.
- Make sure you do not have a memory leak

(10 pts) **Program Style/Comments**
In your implementation, make sure that you include a program header in your program, in addition to proper indentation/spacing and other comments! Below is an example header to include. Make sure you review the style guidelines for this class, and begin trying to follow them, i.e. don't align everything on the left or put everything on one line!
http://classes.engr.oregonstate.edu/eecs/spring2017/cs162-001/162_style_guideline.pdf

```
/*********************************************************
** Program: run_stats.cpp
** Author: Your Name
** Date: 04/08/2017
** Description:
** Input:
** Output:
*********************************************************/
```

Electronically submit your C++ program (**.h, .cpp, and Makefile files**, not your executable) and your test files as a tarred archive by the assignment due date, using TEACH.

You must tar these files together using the following command:

tar –cvf assign1.tar facts.h facts.cpp run_facts.cpp Makefile

****NOTE:** The easiest way to upload your program from ENGR to TEACH is to map a network drive to your home directory on ENGR. Mac or Windows, See:
http://engineering.oregonstate.edu/computing/fileaccess/