TCSS333
C for System Programming
Programming Assignment 3
Most Popular Names (Exercise in Pointers & Arrays)

DUE: See Canvas Programming Assignment 1 link.

You are to submit a single .c file to Canvas and nothing else (do not zip anything, no Eclipse projects, etc.)

This assignment requires the use of files stored in "names.zip." Each file represents a different year (range: 1880-2013) and contains names given to babies born in that year. It also contains the gender (F/M) and the number of babies given that name (each line contains this data for a given baby). The files start with all female names (most popular first) and ends with males (most popular first). For example, the first two lines in the file yob2000.txt are:

Emily,F,25952
Hannah,F,23071

Which indicates there were 25952 females given the name Emily and 23071 females given the name Hannah. As the first two entries, Emily was the most popular and Hannah was the second most popular names given that year.

We will only concentrate on the **100 most popular** **female names** in the years **1920**, **1930**, **1940**, ..., **2010**.

Your program should read this data and create a single summary report file called "summary.csv". Each line of the summary file should contain a single name followed by its popularity rank in each of the 10 annual reports. The name and all the ranks (ranking of 1 as most popular to 100 as least popular) should be separated by commas. The file extension "csv" stands for **c**omma **s**eparated **v**alues and Excel will automatically separate the values into columns when you view such a file as a spreadsheet. **The names in the summary file must appear in alphabetical order**. **For any given name, the ranks should appear in chronological order** (first the rank in 1920, then 1930, and so on until 2010). A name will appear in the summary report if it was one of the 100 most popular names at least once in any of the 10 years being analyzed. A name that appears in the top 100 for one year may be missing from the top 100 in other years. A name not included in the top 100 has no rank at all for that year. You can indicate this as a missing value to Excel by placing two adjacent commas in the summary file.

For example, if your summary file were designed as such (note: you will have other years instead of "etc."):

Name,1920,1930,1940,etc.,2010,

Ann,5,4,3,,1,

Beth,1,2,3,,5,

Cathy,8,5,3,,7,

Excel will open a spreadsheet looking like this:

	A	B	C	D	E	F	G	H	I	J	K
1	Name	1920	1930	1940	1950	1960	1970	1980	1990	2000	2010
2	Aaliyah										56
3	Ann	49	41	35	45	50	62				
4	Beatrice	41	76								
5	Ella	88									13
6	Sarah	50	70	60	88		65	5	6	5	28

Note: The file has column headings and blanks (or no data) for years where the name was not in the top 100.

Continued…

You must use a 2D array of char to store the names. You must use a 2D array of integers to store name ranks. The two arrays are parallel. The first name in the 2D array of chars is associated with the first row in the 2D array of ranks. The columns of the rank array correspond to the 10 years 1920, 1930, ...2010.

Repeat the following until all top 100 names of all 10 files have been processed: Read a name and rank from one of the annual reports. Find this name in your array of names (or add the name if you've never seen it before). Go to the matching row and column (which indicates the given year) of the rank array and enter the name's rank, i.e. the rank is the location within the 100 names (ranking 1-100).

Once you've collected all the data, sort the two 2D arrays to make the names alphabetical. (When you move a name to a new position, remember to move its rank data also!) Print out the contents of the two 2D arrays to the summary file. As you work with the names, make good use of string functions (strcmp, strcpy, ...).

Programming requirements:

- No global variables. No structs! No while (1)! No exit(0)! No break (except in switch)! No goto! No continue! No multiple returns within a function!
- Use #define symbolic constants to establish the maximum length of a name and the maximum number of names (You decide what limits will work.)
- In addition to main, write separate functions for each of these:
  - ✓ function that will read all 10 input files (partly by calling other functions)
  - ✓ function that will read in 1 input file (partly by calling ...)
  - ✓ function that will process a single name and rank pair for some year (e.g. Mary, 82,1990)
  - ✓ function that will sort the arrays (use an N log N algorithm and don't forget, we are dealing with parallel arrays. When a swap is made, the equivalent rankings will also need swapping. As such, the sort function will need access to both, ergo you need to write the sort function)
  - ✓ function that will create the output file
  - ✓ Use other functions as needed. **For all your functions, the array parameters must be declared as pointers.** (This rule is somewhat arbitrary, but the goal is to force you to explore C's way of looking at arrays as pointers!). For example, you will lose points if you have a function signature like this: int findMin(int arr[ ], int size); Instead do this: int findMin(int *arr, int size); This rule applies only to parameter declarations. Inside your functions, you may use as many square brackets as you like, but consider using pointer notation to get more familiar with it. It may be beneficial to look at the file "arraysOfStrings.c" which demonstrates this concept in a simple example.
- Basic Code Quality Requirements (This reminder will not be posted again but, is required in all future assignments)
- Your code must include your name in a comment at the top of the program! Write multiple functions that each do one simply described thing. If it's not possible to see all the code of a function at once (without scrolling), the function is probably too long. The names you choose for variables, functions, etc. should be meaningful and descriptive. Redundant code is code that is clearly repetitive and doesn't need to be. If you find yourself copying and pasting while coding, you very likely are writing some redundant code. Redundant code should be cleaned up before submission. Use functions and loops to eliminate redundancy. Your code must be neatly and correctly indented. Don't use the tab key to indent. Tab doesn't have the same effect in every editor, so code that is beautifully indented in one editor can look bad in another. The text book is a good model for indenting

Once you have generated output, feel free to post samples of your summary data (NOT YOUR PROGRAM CODE!!!!) to the Canvas discussion board. Hopefully a consensus about the correct output will develop.