TCSS 333 HW3 Due: by midnight Friday April 24 (or with 10% late penalty, by midnight Saturday April 25)

Submit only a single .c source file.

Download the file "names.zip". (The files comes from http://www.ssa.gov/OACT/babynames/limits.html) This zip file contains a collection of annual reports stretching back into the 1800's. Each annual report includes the first names given to babies born in that year together with the number of babies who were given that name. The names are in order by popularity, the most popular being listed first. For example, if you look in the file yob1990.txt, the first name listed is Jessica and that name was given to 46,463 babies born in the US during 1990. Jessica was therefore the #1 most popular name in that year. The files actually contain two lists, first the names of girls, then the names of boys. For simplicity, we will focus on the girls' names and ignore the boys' names. Also we will look only at the first 100 most popular names in any given year. Furthermore we will use only the 10 years: 1920, 1930, 1940, ... 2010.

Write a program that reads in 10 annual reports for the specified years and generates a single summary 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 should be separated by commas. The file extension "csv" stands for comma separated values 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. 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 looks like this:

```
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
1	Name	1920	1930	1940	etc.	2010	
2	Ann	5	4	3		1	
3	Beth	1	2	3		5	
4	Cathy	8	5	3		7	
5							

Your summary file should begin with a line containing column headings.

This assignment is largely an exercise in the usage of arrays. 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 is 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.

How to solve it: 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 of the rank array and enter the name's rank. 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!

- Use #define 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:

  - a function that will read all 10 input files (partly by calling other functions)
  - a function that will read in 1 input file (partly by calling ...)
  - a function that will process a single name and rank pair for some year (e.g. Mary, 82,1990)
  - a function that will sort the arrays
  - a 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.

Once you have generated output, feel free to post samples of your summary data to the Canvas discussion board. Hopefully a consensus about the correct output will develop.

Basic Code Quality Requirements
(These guidelines apply to this and all other assignments, but I won't repost them in future homework descriptions.)

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