C/C++ Programming Homework #8 April 26, 2014 (Sat. at 11:59 PM)

(Note: This homework can be used to replace a previously unsubmitted homework or be counted as extra credit if you can completed all homeworks.)

Your goal is to design and write a menu-driven program that provides the user the ability to search a large (maximum size of 1,000) collection of customer records. The customer records are in the file customerData.txt. **Each customer record is on a single line with 12 fields separated by commas(‘,’).** The order of the fields on a line is: First Name, Middle Initial, Last Name, Street Address, City, State, Zip Code, Country, Email Address, Telephone Number, Gender, and Birthday.

Your goal is to design and write a menu-driven program that provides the user the ability to:

- search for customer records satisfying a user specified criterion (e.g., State is “IA”). After each search criterion is specified, the number of customers satisfying the search should be displayed to the user. The search process should repeatedly allow the user to specify one criterion at a time to narrow the search (e.g., State is “IA” and Gender is “female”).
- save search results to a new text file in one of two possible formats (as mailing labels or formatted just like the customerData.txt file). Allow the user to enter the file name for saving.
- update a customer record (e.g., a customer might have moved or gotten a new phone #) in the file of records, customerData.txt. (You might be able to reuse some of your searching functions to help find the customer record being changed)

On startup **before** your program enters the menu-driven phase, it should:

- read each customer-record line in customerData.txt as a C++ string,
- Using C++ string member functions (e.g., find, length, etc.), split the line into a smaller C++ strings and assign them to fields in a Customer structure. The Customer structure should contain members (firstName, middleInitial, lastName, etc.) all of type `string`,
- store this struct for a customer into an array of customers (i.e., you’ll have an array of structures)

During the menu-driven phase, searches and updates should be on array(s) of customers. When the user selects from the menu that they want to exit the program, then the customers array should be written back to the customerData.txt file if any customer records have been updated. (When debugging your program, you should probably write to a different file.)

If you save the search results as mailing labels, each mailing label should be formatted as below with 2 blank line separating each label (one blank line above and one below each mailing label):

Jane Smith
123 Main Street
Cedar Falls, IA 50613

Additional Requirements:

Be sure to divide the program into functions that perform each major task.

When you write your program, be sure to use general conventions of good style:

- use meaningful variable names with good style, i.e., useCamelCase (or use_underscores) (I like to declare them one per line with a following comment if necessary)
- use meaningful named constants (e.g, PI, STATE_SALES_TAX) where appropriate with good style (ALL_CAPS_AND_UNDERSCORES). Put your global constants where they can be found and changed

easily in future versions of your program, e.g, after your initial comments describing the program and before your main function definition.

- use comments at the start of the program, before each function, and before any especially difficult section of code to understand (I like to label the closing set bracket, '}', with some indication of what's being closed)
- place the `main` function near the top of the program with user-defined functions below it
- use *white space* (spaces, indentation, blank lines) to make you program more readable by:
  - aligning the opening set/curlly brace, '{', with the corresponding closing '}' one (I like to put the '{' on the same line with the programming construct (e.g., main function definition) with the closing '}' aligned with the start of the construct)
  - indent all the lines inside a set of of braces
  - you blank lines to separate logical units of the code, e.g., between variable declarations and executable statements

The data file for the homework is at:
http://www.cs.uni.edu/~fienup/cs1160s14/homework/customerData.txt

This file starts with the following lines:

```
First Name,Middle Initial,Last Name,Street Address,City,State,Zip Code,Country,Email Address,Telephone Number,Gender,Birthday
Woodrow,C,Wilson,2362 New Street,Eugene,OR,97408,US,Woodrow.C.Wilson@spambob.com,541-337-9453,male,11/26/1984
Eric,A,Stutler,568 Nuzum Court,East Aurora,NY,14052,US,Eric.A.Stutler@trashymail.com,716-652-4943,male,11/24/1947
David,E,Herbert,3678 Seltice Way,Boise,ID,83704,US,David.E.Herbert@spambob.com,208-703-8246,male,2/16/1971
Lee,A,Andrews,3472 Berkshire Circle,Knoxville,TN,37917,US,Lee.A.Andrews@spambob.com,865-566-4125,male,11/15/1973
Rena,D,Adkins,3153 Cardinal Lane,Cleveland Heights,OH,44118,US,Rena.D.Adkins@trashymail.com,216-932-7637,female,1/14/1975
Reyna,R,Brown,2093 Middleville Road,Los Angeles,CA,90014,US,Reyna.R.Brown@pookmail.com,626-388-0030,female,3/20/1942
Duane,E,Hall,3467 Ray Court,WAGRAM,NC,28396,US,Duane.E.Hall@pookmail.com,910-369-3902,male,7/30/1962
Fannie,C,Chapman,1791 Shingleton Road,Bridgman,MI,49106,US,Fannie.C.Chapman@spambob.com,269-465-7477,female,10/5/1961
Maggie,D,Betts,1972 Twin Willow Lane,Fayetteville,NC,28304,US,Maggie.D.Betts@spambob.com,910-425-9414,female,8/16/1980
Glenn,M,Stephens,280 Hillcrest Circle,MAPLE PLAIN,MN,55359,US,Glenn.M.Stephens@dodgit.com,763-479-7845,male,7/18/1985
...
```

The steps for the homework submission system are:

1. Design, write, debug, and test your program in a new hw8 subdirectory on student.cs.uni.edu. When you are ready to submit your homework, copy it back to your computer using FileZilla (or WinSCP) program, zip the whole hw8 folder on your computer by right-clicking on it and selecting `Send to | Compressed (zipped) folder`. This will create a new file called `hw8.zip` which you will submit electronically. (This assumes Windows OS....)
2. Log on to the submission system at: **https://www.cs.uni.edu/~schafer/submit/which_course.cgi**

   (It is very likely that you will get some security certificate warnings when trying to use this. You may add an exception and accept the existing security certificate.) Use the same CatID user-name and password you use to log on the lab computers.
3. Select the course and section number of "CS 1160, C/C++ Programming, Fienup". Click the "Continue".
4. Select the homework that you wish to submit: "HW 8: Customer Search". Click the "Continue" button.
5. Specify how many extra files you want to submit. Just leave it at 0. Click the "Continue" button.
6. Upload your program by Browsing and selecting your `hw8.zip` file. Click the "Continue" button.
7. The next page reports on the status of the upload(s). You can always continue to upload a better version of the program until the deadline. The newer file will replace an older file of the same name.

(If you miss the deadline, you'll need to submit it as above, but select "Late Homeworks" in step 4 above.)