The University of Alabama in Huntsville
Electrical and Computer Engineering
Project 9 (30 points) – 10% bonus for exact match of all program output
Submit Your Solution Using Canvas by Noon on Friday November 2, 2018
➔ Points will be deducted for those solutions with differences in output formatting ←
(late submissions will be accepted from Noon to 2:00PM on 11/2/2018)

<Project 9 Description>

Project 9 introduces the student to the use of structures in programming. This assignment will tie in several of the key areas previously used in the lab assignments. In particular, input from a file, output to a file, functions (value returning and void), if statements, while loops (or some other form of looping) and others. ***You are not allowed to use arrays in this program!!!***

Problem description: You are writing the subscription renewal system for a magazine. Information for each subscriber is kept in a file. The information for one subscriber is called a record, and the input file may contain several records or no records. After reading each record, a test is made to determine if the subscribers' subscription has expired. If a subscribers' subscription is still active (number of months left on subscription is greater than 0), no action is necessary and the next record is obtained. If a subscribers' subscription has expired (number of months left on subscription is 0), information about the subscription is written into an output file before the next record is obtained.

You may run the sample solution **Project_09_solution** by typing the following at a command prompt in a terminal window**:**

/home/work/cpe211/Executables/Project_09/Project_09_solution
(Input files are contained in P9_in.zip)

The comparison script can be run by using the following command
/home/work/cpe211data/Project_09/CompareSolution.bash Project_09.cpp

<Project 9 Directions>

On Project 9, you may only use concepts presented in Chapters 1 - 10 of your textbook!!
The following specific concepts are not allowed at all: Global Variables, Arrays and Recursive Function Calls. All function calls are made from main only.
The input file can be opened one time only

Use of global variables, arrays, or recursive calls will result in zero credit (0).

In addition to the function* main()**, *your solution must include at least the four functions mentioned on item #2 on page 3 of this description to receive full credit. For each of these functions, only one of the parameters may be a structure variable and this structure variable must be a top level structure (see structure information on page 2) variable.

Using your favorite text editor, type your solution and save it as a file named **Project_09.cpp** within your **CPE211_FALL18/Project_09** directory. If there are syntax errors, correct them and compile again. Once your program successfully compiles, run it and verify that the output for your modified program matches the output from the solution executable – **Project_09_solution**.

Once you are satisfied with your solution, submit Project_09.cpp via Canvas.

<Project 9 Structure Details and Other Hints>

Read the entire project handout before writing any code, think about it, read it again, plan out your solution (write a functional decomposition or draw your own flow chart to organize your solution), and then write and test your code.

1. ***There are 5 structure DataTypes that must be declared***. These must be ***global*** struct declarations (place these declarations after the **using namespace std;** statement and before your function prototypes). NOTE: VARIABLES DECLARED AS THE DATA TYPE OF THE STRUCTURE ARE NOT TO BE DECLARED AS GLOBAL VARIABLES. The structures will be written in a hierarchical fashion. One will be the top-level structure and the other four will be sub structures (see pages 486-488 of the text).

The top-level structure has the following 3 members:

subscriberName	(declared as a **Subscriber Name structure**)
address	(declared as an **Address structure**)
renewal_information	(declared as a **Renewal Information structure**)

Each field of the top-level structure is also a structure, creating a hierarchical structure. The details of each nested structure are summarized below – be sure to pay attention to the data types of each structure member!!

The sub structures (nested structures) have the following members:

Subscriber Name Structure:

first name	(a string)
last name	(a string)
customer id	(an integer)

Address Structure:

street number and name	(a string)
city	(a string)
state	(a string)
zip_code	(an integer)

Date Structure:

month	(a string)
day	(an integer)
year	(an integer)

Renewal Information Structure:

number of months left	(an integer)
last renewal notice sent	(a **Date** structure)

NOTE: The order in which you declare your structures is important. The sub structures must be declared ***before*** the top-level structure. Furthermore, the **Date Structure** must be declared before the **Renewal Information Structure**.

2. This program must include the following 4 functions:
   a. A function that opens the input file and includes all error-handling logic. If the user-specified file does not open, this function should use a **LOOP** that will repeatedly
      - output an error message
      - clear (reset) the file stream
        `inputStreamVariable.clear();`
      - reprompt the user and input their next file selection
      - attempt to open the next selection

      until the user-specified file opens successfully or until the user types Control-C to exit the program. ***Remember – Recursion is not allowed!!***

   b. A function that opens the output file and includes error-handling logic similar to that describe above in **Part a** for opening of the input file. The program is to correctly handle an output file that does not open successfully (see the last page of this document).

   c. A function that inputs one complete record at a time from the input file and stores the information in the appropriate structure variable fields (member names). You may assume that all fields are present for each record and that the information in those fields is of the appropriate data type. This function could be a Boolean value returning function. **This function cannot have an ofstream parameter**

   d. A function that writes a subscriber record to the output file if and only if that subscribers' subscription has expired (number of months left is 0). Run the provided sample solution to obtain a sample of what is to be written to the output file. **This function cannot have an ifstream parameter**

3. For the functions above only one of the parameters can be a structure variable, and this structure variable must be a top level structure variable. **Furthermore, there can be at most 3 parameters for any function ←**

4. The function main() should contain the loop that controls the reading of the records. In main(), two counters are used to keep track of (1) the number of subscribers present in the file and (2) the number of subscribers that have expired subscriptions. **After the last subscriber record has been processed, a message is written to the terminal indicating how many subscribers were in the file and how many subscribers have expired subscriptions**.

5. ***Since there are multiple records possible in an input file and you cannot use arrays – even if you know how, each record is read from the input file, processed, and written to the output file (if necessary) before the next record is read.***

6. ***Your program will not know in advance the number of records that exist within a file.*** An example record of the information in the input file is arranged as follows (NOTE: the information shown as a comment is not present in the input file):

```
Joe                 // First name
Smith               // Last Name
14235               // Customer id
123 Abbey Way       // Street Number and Name (must be read with getline)
Huntsville          // City (must be read with getline)
Alabama             // State (must be read with getline)
35789               // Zipcode
0                   // # of months remaining on subscription
March               // Last renewal notice month
10                  // Last renewal notice day
2002                // Last renewal notice year
```

7. Sample input files have been provided to the student for testing of the program. The program will be tested with different input files (not available to the students). The format of the input file used for testing is identical to the format of the sample input files.

   Reading information from the input file will require statements like the following:

```
inputStreamVariable >> structVar.name.first;
getline (inputStreamVariable, structVar.address.city,'\n');
inputStreamVariable.ignore(INT_MAX,'\n');
```

   It is up to the student to determine where, when, and how to use these statements. Note: The above input statements illustrate how information can be read directly into the member of a structure.

8. If an input file does not contain any records, it will be empty. **For this input file, a message is written to the output file and to the terminal**. Run the provided sample solution with the file P9_in4.txt to see the required messages and program operation.

9. When reading in the information for a subscriber, read the first name and then test the input stream to see if it is in the fail state. If it is, do not read any more information from the input file, and then perform any necessary actions based on reaching the end of the file.

10. It is easiest to handle a "empty" input file by using a priming read action before entering a while loop. The priming read in this case will be a read of an entire subscriber record. The file stream can be tested after the priming read (and before the while loop) to see if the eof status bit is true ➔ empty file

11. Remember that input and output streams used as parameters in a function must be reference parameters.

12. When processing a record for a subscriber, print out the phrase "Processing Customer Id: ####" right after the customer id has been read.

13. To test a bad output file, use a file name of Bad/Input/File.txt or any name you want as long as it has a / somewhere in it.

14. To determine the input order and the expected output, run the sample solution **/home/work/cpe211/Executables/Project_09/Project_09_solution**

15. Match your output as closely as possible to the output of the sample solution.

16. Don't forget to run the comparison script to see how your solution compares to the sample solution. **/home/work/cpe211data/Project_09/CompareSolution.bash Project_09.cpp**