Assignment #6: Snack **Due:** Wed. Oct. 11th **(midnight)**

Program files:

Code:	snack_driver.cpp
	snack.h
	snack.cpp
Input:	snack_data.txt
Output:	a6.txt

Objective:
For this assignment you will use the **Snack** class that is being provided for you. You will download the class files from Canvas. You will be adding additional functionality to the class and writing a driver file (main) to utilize the class.

Program Overview:
This program will utilize a C++ class that is capable of holding information for a single **Snack.** A Snack may be a fruit, vegetable, or a processed food. The data file, **snack_data.txt** contains data for 40 snacks. You will load the information from file into an array of Snacks. Once the snack data is loaded you will then display a menu asking the user what information they would like from the data in the array. Responses to the user requests should go to both the console AND the output data file.

You will also be breaking down this assignment into multiple code files. The code files you will have for this assignment include:

- snack_driver.cpp
  - This will contain your general function prototypes and main. This also holds the function definitions after main.
- snack.h
  - This will contain your snack declaration only
- snack.cpp
  - This will contain your snack member function definitions only
- snack_data.txt
  - This contains your input data that will populate your array
- A6.txt
  - This contains your results of your program

Class Modifications
Before you start working on your driver file, you will need to complete **snack.cpp**. It is your responsibility to complete each of the function definitions.

You will also need to **ADD** a **copy constructor** to the class. This means you will need to add the prototype and the class definition. At this time, neither exists in the class.

Snack_data.txt
You will also need to **ADD** a field to the text file for each of the snack objects. Each object is missing the **character** field that represents the snack type. You will need to add a single character (F, V, P) for each snack. It is up to you where you add it. Make sure you accommodate your read function so it can read this value also.

Program Requirements:
First, your program may <u>NOT</u> contain any global variables. Main() will be responsible for calling functions. Therefore, main() will be a series of function calls along with any necessary variable declarations you will need for you program.

main()
Your main() function should perform the following tasks:

- Declare any variables you will need for your program, including the array
- Declare and open two filestreams (one for reading, one for writing)
- Loop x number of times
    - Read a Snack
    - Load the array, using the next available array location
- End loop
- Close input filestream
- Loop
    - Call displayMenu
    - Perform the appropriate operation via a function
    - Display the result (console and output file)
- End loop
- Close output filestream
- End program

- **void readData(Snack &s, fstream &inFile)**
    - This function will accept an empty **Snack** object by reference and load it with the next snack's information from file. The filestream is received by reference.
    - You will need to read one line at a time from file, and then call the appropriate member function to set the data member
    - DO NOT close the file

- **void loadArray(Snack s, Snack snackArray[], const int SIZE, int &current)**
    - This function will accept the most recent Snack object just read from file, the array of Snack objects, an integer that represents the size of the array, and the current array index.
    - The current array index is what should be used when inserting into the array.
    - After inserting the object into the array, you will need to increment current.

- **void displayMenu(Snack snackArray[], const int SIZE, fstream &outfile)**
    - This function will display a menu to the user and ask them what they would like to do
    - You will keep displaying the menu until the user choses to "quit"

- Once the user makes a menu selection, you will call a function to complete that specific task
- Menu options include:
    - Display all Fruits
    - Display all Vegetables
    - Display all Processed Snacks
    - Display the Lowest in Calories
    - Display the Highest in Calories
    - Display a Specific Food by Name
    - Quit

I am giving you control over how you write your functions for your menu options. With this said, you could write a separate function for each menu option <u>OR</u> you could combine some IF you send the appropriate arguments to the function. For example, you could combine the first three options into one function and send a flag that indicates if you are to display a fruit, vegetable, or a processed food.

The same for the snack with the highest and lowest caloric count.

Displaying a specific food will need to be separate as will quitting the program.

You may use letters or numbers for the menu. Numbers tend to be easier. Also a switch statement works nicely on determining the menu choice.

Running Your Program
You need to run your program so that you eventually select every menu option. Your output data file should reflect only the output from the user's request.

Please make sure you include the appropriate program header information at the top of the first page of your program (see the department guidelines). When you are finished, compile, and then run the program. If it does not compile, fix any errors and try again. <u>When your program is working, submit your **snack_driver.cpp, snack.h, snack.cpp** files along with your output file, **a6.txt** by midnight on **WEDNESDAY, OCT. 11TH.**</u>