CSCE 2014 – Bonus Programming Project (20 points)
Due Date – 10/10/2022 at 11:59pm

1. Problem Statement:

The purpose of this project is to give students more experience reading and writing text files and storing and processing information in vectors of objects.

You will be given an input file containing 10 pieces of information about recent real estate listings (house_number, street_name, city, state, zip_code, asking_price, number_bedrooms, number_bathrooms, house_size, lot_size). This file has the following format:

```
792 W Foothills Dr, Fayetteville, AR 72701
389000
5 bed
2.5 bath
2562 sqft
0.29 acre lot
```

Notice that five pieces of information are included together on the first line separated by spaces or commas. The remaining five data values are on separate lines and have zero or more descriptive words after the data value. The input file will have a blank line between real estate listings to make it easier for humans to read.

Your program should read the input file and store this information in a **vector** of RealEstate objects. Once this is done, your program should loop over the vector and produce an output file in "comma separated value" format. Information describing each RealEstate object should be written on separate lines in the output file in the following format:

```
792,W Foothills Dr,Fayetteville,AR,72701,389000,5,2.5,2562,0.29,
```

Notice that there are no spaces before or after the commas, and there is a comma at the end of the line. In addition, the descriptive words have been removed. This format should be easy to read into a spreadsheet.

2. Design:

Your first design task is to create a C++ class called RealEstate to store the 10 pieces of information above in private variables with appropriate data types. Your class should have a default constructor, copy constructor, destructor. You can implement 10 get/set methods for these data fields OR you can implement get/set methods with 10 parameters. You do not need to perform error checking on the data values to ensure they are valid.

Your second design task is to work out the logic for how to read real estate information from an input file and store it in a vector of RealEstate objects. The goal is to store the data in the vector in the same order that it appears in the input file. There are several ways to do this using the constructor functions and get/set methods you created above. Another option would be to add a "read" method to the RealEstate class that takes an open filestream object as a reference parameter, and reads real estate data from this file.

Your third design task is to loop over the vector and produce an output file in "comma separated value" format described above. One option would be to use the get/set methods to extract the 10 pieces of information to be printed. Another option would be to add a "print" method to the RealEstate class that takes an open filestream object as a reference parameter, and writes real estate data to this file.

3. Implementation:

To implement your project, you should break your code down into **three** files: real_estate.h, real_estate.cpp, and main.cpp. You are strongly encouraged to look at programs on the class website for sample code to assist in the implementation of this project.

As always, it would be a good idea to start with "skeleton methods" to get something to compile, and then add the desired code to each method incrementally writing comments, adding code, compiling, debugging, a little bit at a time. Once you have the methods implemented, you can create a main program with s simple menu interface that calls these methods to complete your project.

Remember to use good programming style when creating your program (good names for variables and constants, proper indenting for loops and conditionals, clear comments). Be sure to save backup copies of your program somewhere safe. Otherwise, you may end up retyping your whole program if something goes wrong.

4. Testing:

For this project students are required to compile and test their program on Linux. Once you have logged into turing.csce.uark.edu, copy your source code into a folder called "bonus". Go into this folder and type in the command "script". This will create a "typescript" file that records all user input and program output until you type "exit".

Compile your program using "g++ -Wall *.cpp -o main.exe". Then run your program by typing "./main.exe". Enter a sequence of commands to verify that your program operates correctly for all of the requirements listed above. Also check for the error handling capabilities of your code.

When you are finished testing your program, type “exit”. Copy the “typescript file back to your personal computer to be included with your code and project report when you submit your completed project into Blackboard.

5. Documentation:

When you have completed your C++ program, write a short report using the project report template describing what the objectives were, what you did, and the status of the program. Does it work properly for all test cases? Are there any known problems? Save this report to be submitted electronically.

6. Project Submission:

In this class, we will be using electronic project submission to make sure that all students hand their programming projects and labs on time, and to perform automatic plagiarism analysis of all programs that are submitted.

When you have completed the tasks above, copy all of your source code, your typescript file, and your project report into a folder called “bonus”. Compress this directory into a single ZIP file called “bonus.zip”, and upload this ZIP file into Blackboard. The GTAs will download and unzip your ZIP file and compile your code using “g++ -Wall *.cpp” and run your program to verify correctness.

The dates on your electronic submission will be used to verify that you met the due date above. All late projects will receive reduced credit:

- 10% off if less than 1 day late,
- 20% off if less than 2 days late,
- 30% off if less than 3 days late,
- no credit if more than 3 days late.

You will receive partial credit for all programs that compile even if they do not meet all program requirements, so handing projects in on time is highly recommended.

7. Academic Honesty Statement:

Students are expected to submit their own work on all programming projects, unless group projects have been explicitly assigned. Students are **not** allowed to distribute code to each other, or copy code from another individual or website. Students **are** allowed to use any materials on the class website, or in the textbook, or ask the instructor and/or GTAs for assistance. This course will be using highly effective program comparison software to calculate the similarity of all programs to each other, and to homework assignments from previous semesters. Please do not be tempted to plagiarize from another student.

Violations of the policies above will be reported to the Provost's office and may result in a zero on the programming project, an XF in the class, or suspension from the university, depending on the severity of the violation and any history of prior violations.