CS 202 - Computer Science II

Project 3

Due date (FIXED): Wednesday, 2/15/2017, 11:59 pm

Objectives: The two main objectives of this project is to test your ability to (1) create and use pointers, and (2) create and use C++ classes. A review of your knowledge of structs, arrays, iostream, file I/O and C-style strings is also included.

Description:
This project will expand Project 2 by adding additional functionality, using pointers, and implementing abstract data types (ADTs) through classes. **Pointers must be used for all array manipulation**, including arrays with ADTs (structs, classes) e.g, rental cars, rental agencies. **Pointers must be used in function prototypes and function parameter lists** - not square brackets. Make sure you change your string functions (e.g. string copy) using pointers. Square brackets should be used only when declaring an array. **Pointers can only be moved by incrementing or decrementing** (i.e., ++ or - -), or by **setting the pointer back to the base address** using the array name. All pointers must be passed by value. You may freely use the arrow operator (->) with any pointers if you so wish.

The additional functionality is as follows: You are given an updated data file where there are 3 rental car agency locations, where **each of the 3** locations (**RentalAgency**) has **5** cars (**RentalCar**). You will have the **same menu options**, but the **functionality has been updated** below. Note: using multiple helper functions to do smaller tasks will make this project significantly easier. You may want to create a function that will get a car from a location based on the location and car indices.

The RentalCar Class will contain the following data members:

- **m_make**, a C-style string
- **m_model**, a C-style string
- **m_year**, an int
- **m_price**, a float (price per day)
- **m_available**, a bool (1 = true; 0 = false; try to display true/false using the "boolalpha" flag)

and will have the following methods:

- **Default Constructor** – will set the aforementioned data members to default initial values.
- **Parameterized Constructor** – will create a new object based on the values passed into it.
- **Get and Set methods** for all data members.
- **Print** – will print out all the car's data.
- **EstimateCost** – will estimate the car's cost *given* (a parameter passed to it) a number of days to rent it for.

The RentalAgency ADT will be struct and will contain the following data members:

- **name**, a C-style string
- **zipcode**, an int array of size 5
- **inventory**, an array of RentalCar objects with a size of 5

The menu must have the following updated functionality:

- **Read ALL** data from **file** (the file has been **structured** where the first line is the car agency info, followed by 5 cars).
- **Print out ALL** data for **all agencies** and **all their corresponding cars** in a way that demonstrates this relationship (see Sample Output section).
- **Estimate** car **rental cost** - **prompt** for an **agency** (e.g., Hertz) and **car number** (where 1-5 are the cars at each agency).
- **Find** the **most expensive** car - print the single most expensive car out of all 3 Agencies.
- **Print out** only the **available** cars - from **all agencies**.
- **Exit** program.

The following minimum functionality and structure is required:

- Ask the **user** for the **input file** name.
- The list of cars must be stored in an **array of objects**.
- Use **character arrays** (i.e., C-style) to hold your strings. No string data type!
- Write **multiple functions** (Hint: each menu option should be a function).
- At least on function must use **pass by reference**.
- **Pointers** must be used for **all array manipulation**.
- **Pointers** must be used in **function prototypes** and **function parameter lists** (not brackets).
- **Pointers** can only be **moved by incrementing or decrementing**:
  double d[3] = {1,2,3};
  double* d_Pt = d;
  for (int i=0; i<3; ++i,++d_Pt){ cout<<*d_Ptd; }
- Or by **setting** the pointer **back to the base address** using the array name.
  d_Pt = d; cout<<endl<<*d_Pt<<endl;
- Write your **own string copy**, **string compare** (or other) functions as needed, modify any existing versions to account for the updated **specification** about using **pointers**.
- The other functionality and structure of the program should remain the **same as Project #2**, including **writing to screen** and **file** and **restrictions on string** libraries, **global variables** and **constants**, etc.
- For this, and hence for any other Project of similar requirements, **"Print out"** implies **Console** and **File Output** (the same information, but to a different file (with a characteristic name, e.g. "AllAgenciesAllCars.txt") for each different implemented functionality.

Implement the concepts of encapsulation and data hiding as much as possible!

This is a chance to experiment as much as possible with classes, and their concepts as taught in class. It is not a strict requirement that all RentalCar data members are private at this point. Try your best in order to acquaint yourself with these new concepts at this early point, so that it pays off in future project which will impose such hard requirements.

Sample Output for menu option 2:

Hertz 93619

2014 Toyota Tacoma $115.12 per day Available: true

2012 Honda CRV $85.1 per day Available: false

2015 Ford Fusion $90.89 per day Available: false

2013 GMC Yukon $110.43 per day Available: false

2009 Dodge Neon $45.25 per day Available: true

Alamo 89502

2011 Toyota Rav4 $65.02 per day Available: true

2012 Mazda CX5 $86.75 per day Available: true

2016 Subaru Outback $71.27 per day Available: false

2015 Ford F150 $112.83 per day Available: true

2010 Toyota Corolla $50.36 per day Available: true

Budget 93035

2008 Ford Fiesta $42.48 per day Available: false

2009 Dodge Charger $55.36 per day Available: true

2012 Chevy Volt $89.03 per day Available: false

2007 Subaru Legacy $59.19 per day Available: false

2010 Nissan Maxima $51.68 per day Available: true

The completed project should have the following properties:

- Written, compiled and tested using Linux.
- It must compile successfully using the g++ compiler on department machines.
  Instructions how to remotely connect to department machines are included in the Projects folder in WebCampus.
- The code must be commented and indented properly.
  Header comments are required on all files and recommended for the rest of the program.
  Descriptions of functions commented properly.
- A one page (minimum) typed sheet documenting your code. This should include the overall purpose of the program, your design, problems (if any), and any changes you would make given more time.

Turn in: Compressed .cpp file and project documentation.

Submission Instructions:

- You will submit your work via WebCampus
- Name your code file proj3.cpp
- If you have header file, name it proj3.h
- Compress your:
  1. Source code
  2. Documentation

  Do not include executable
- Name the compressed folder:
  PA#_Lastname_Firstname.zip
  Ex: PA3_Smith_John.zip

Late Submission:

A project submission is "late" if any of the submitted files are time-stamped after the due date and time. Projects will be accepted up to 24 hours late, with 20% penalty.