Assignment 7 – December 8, 2022 at 11:59pm
This assignment must be done individually.

***** NO LATE WORK IS ACCEPTED ON THIS ASSIGNMENT. ALL WORK MUST BE TURNED IN BEFORE READING DAY (DECEMBER 9) – THEREFORE, ALL WORK IS DUE AT 11:59PM ON DECEMBER 8 *****

1. Problem Statement

The goal of this programming assignment is to further your experience creating classes, working with file input and output, and arrays using user-defined data types. You are tasked with constructing Donna's stand at the local farmer's market, printing data to the user and performing actions from a list of options as they see fit. Students are required to submit their code, a typescript file demonstrating the program compiling and working with different test cases, and a report detailing what decisions were made and why.

2. Description

Donna has had a tumultuous yet gratifying career, ranging from being a connoisseur of Italian American cuisine to being thrown in jail for theft. After years of heartache and triumph, she has decided to retire and return to her roots as a farmer. However, Donna just recently made it out of jail and still owes Olive Garden a fair bit of money. To pay off her debt, she's decided to sell most of her animals at the local farmer's market.

Your job is to help Donna create her stand downtown, displaying the animals she has for sale to the user and which animals the user would like to see given a set of parameters. Donna has a list of the animals she'd like to sell, but it's full of unwanted data! We need to find a way to process Donna's inventory to create a new Animal object for each animal that's read in the inventory.

After processing the data, we must ask the user what they'd like to do at the stand. You should be performing actions and making calls to methods based on the user's choice until the user decides to leave the program. The user will then receive a receipt with their total and be on their merry way.

3. Implementation

Download the provided code that has your readFile, writeFile, printMenu, and getUserChoice functions already written for you!

Task 1: Create the Animal Class

This class should store the basic information about an animal found inside the data set. The following are the class's variables:

- ID (this number is generated when reading in the file)
- Type
- Weight
- Age
- Price

Notice that ID is not a part of the data set but should still be included in the class. Each animal that is read from the file should be assigned an ID corresponding to the order they were read in (e.g. the first animal read will be declared with an ID of 1, second animal with 2, so forth). This will be used for option 6 in the menu.

This class's methods should consist of:

- Default constructor
- Setter constructor
- Copy constructor
- Destructor
- Getter methods for each class variable
- Setter methods for each class variable
- Print method to print contents of each class variable

Note: the getters are used in writeFile, which is provided – be sure you have chosen the same names and capitalization for these functions!

Task 2: Create the AnimalDB Class

This class is a "container class," meaning that it's a container of Animal objects (see lab 12 and the Course class; AnimalDB stands for "animal database"). There should be 2 arrays declared for this class: one that stores the contents of Donna's inventory (data read in from the file), and one that stores the user's current choices. Both arrays should be of Animal data type. You will need two more variables for this class that track the number of animals being stored inside each array respectively.

Each array inside this class should be declared with a global variable denoting the maximum number of animals, which is 100.

This class's methods should consist of:

- Default constructor
- Copy constructor
- Destructor
- Getter methods for variables tracking the size of each array
- Method for returning an Animal object at a given index in the array for the user's shopping cart
- Method for returning an Animal object with a given ID in the array for Donna's inventory
- Method for returning the current running total of the user's shopping cart
- Insert method for adding a new animal to Donna's inventory
- Insert method for adding a new animal to user's shopping cart
- Print method for Donna's inventory
- Print method for user's shopping cart
- Methods to print Animals based on user-specified parameters (type, age range, etc.)

These methods will require error checking to ensure that no Animals are being inserted into the arrays if the maximum size has been reached, check to see whether any animals were printed within the desired range, etc. Most of your methods should return a Boolean value based on whether they were executed successfully. You should print an error message at the place of the function call if it did not.

Task 3: Test the read file method in main

This function takes in 2 parameters: a string representing the file name, and an AnimalDB object passed by reference. The AnimalDB object is needed since we'll be making calls to a method inside the AnimalDB class to insert a newly defined Animal object into Donna's inventory.

Test that this method is working properly by making a call to the print method you implemented inside the AnimalDB class with the object you passed to the read method. Remember that we make calls to methods inside of a class like so:

```
SomeClass myObject;
myObject.someMethod();
```

Task 4: Examine the user menu

Look over the provided code for the user menu that is printed to the user before they're prompted for input. Error checking is provided so that the user's input is within a valid range. The user is presented with the following options:

1. Print all animals in Donna's inventory
2. Print user's current shopping cart
3. Print animals of a certain type
4. Print animals within a range of weight
5. print animals within a range of age
6. Add animal to your shopping cart
7. Print current shopping cart total
8. Complete order, print receipt and exit program

We will be using the AnimalDB object you passed to the read function to make calls to methods inside the AnimalDB class based on the user's selection. For example, if the user were to input 5 as their choice, we would make a call to the method implemented inside the AnimalDB class that prints animals within a user's desired age range. Remember, if no animals were printed to the user, print a message stating that there are no animals that fit within the specified arguments.

This printing of the menu and gathering user choice should be placed in their own respective functions, and the process should continue looping until the user decides to end the program.

Task 5: Test the write file method

Once the user decides to leave the program, we call on the write method to print their receipt. This method writes the contents of each animal in the user's shopping cart to a CSV (comma separated values) file. The method takes in two parameters: a file name to open and the AnimalDB object being used throughout the rest of the program. Because we're utilizing the size of the user's shopping cart, we use the object to make a call to the getter for the current number of animals inside the cart.

There's also a method inside the AnimalDB class you should have implemented that returns an Animal at a given index in the array for the user's shopping cart. The Animal object this method returns is used to write its contents to the file. Each data field is separated by commas as denoted by the file extension.

The last data field, the price of each animal, has an extra row for printing the total/sum of the user's visit. To test that this is working properly, open your CSV file in Microsoft Excel to see if the data fields are placed in their corresponding columns and that the user's total is aligned with the other prices.

4. Testing

Your program must run using the g++ compiler on Turing. You should test your program with many inputs and scenarios to verify that it's working properly. You can develop your program in OnlineGDB as well – there shouldn't be any issues with the data set's file size being too large.

Because error checking is heavily used throughout this program, **use many test cases for your typescript submission**.

To copy over your program files from your local machine to Turing to run your typescript, you can use Filezilla (do step 2, then start on step 4 below after copying your files over), copy-paste each individual file (do step 2, then start on step 4), or the scp command like so:

1. Open new terminal/command prompt window
2. Open another terminal/command prompt window, log in to Turing
   a. ssh username@turing.csce.uark.edu
3. If copying over entire folder, from local machine window run:
   a. scp -r my_folder username@turing.csce.uark.edu:~/
   b. will copy folder to home directory in Turing
   c. omit the -r flag to copy over single file
4. From Turing window, cd into folder you just copied over
5. Run this command to start typescript
   a. script lastname-HW7.txt
6. To compile
   a. g++ -Wall *.cpp -o HW7.exe
   or
   b. g++ -Wall main.cpp AnimalDB.cpp Animal.cpp -o HW7.exe
7. To run, type:
   a. ./HW7.exe
8. Test your code thoroughly while your typescript is running
9. To finish typescript, type:
   a. exit
10. To copy typescript from Turing back to local machine, run this command from local machine window (cd into where you want the file to be first; preferably desktop):
    a. scp username@turing.csce.uark.edu:~/folder_you_ran_typescript_in/lastname.txt .

5. Documentation

When you have completed your C++ program, write a short report using the "Programming Project Report Template" describing what the objectives were, what you did, and the status of the program. Does your program work properly for all test cases? Are there any known problems? Save this project report in a separate document 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 go to Blackboard to upload your files. For full credit, you need to submit the following items:

1. **Three C++ (.cpp) files**: hw7.cpp *(or main.cpp)*, AnimalDB.cpp, and Amimal.cpp
   Please do not submit code in .docx, .txt, or any other format aside from .cpp!
2. **Two Header (.h) files**: AnimalDB.h, Animal.h
3. A **project report** in .docx or .pdf format following the template on the course website.
4. **One typescript file** that includes the compilation of your .cpp files, as well as running the program with multiple tests.

The dates on your electronic submission will be used to verify that you met the due date above. **No late work is accepted for this project.**

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 F in the class, or suspension from the university, depending on the severity of the violation and any history of prior violations.