COMSC-200 Lab 5
Friends

GOOD PROGRAMMING PRACTICES *show / hide*

ABOUT THIS ASSIGNMENT
In this lab session, you begin solving the **ShortestRoute** problem -- a problem that will be completed in the final project. You will also practice programming with classes by doing exercises from our textbook.

As you complete this assignment, post the required files to the COMSC server. You may post them all at once, or one-at-a-time as you complete them, as you prefer. The individual labs are graded in order, starting with "a", which must be fully correct before "b" is graded, and so on. This assignment is worth 50-points, to be awarded after all labs are successfully completed. Use the "Submit your SP2015 work" link on the class website to post your file for this lab, using the **lab5** folder provided.

Lab 5a [`Date.h`, `Date.cpp`, `Employee.h`, and `Employee.cpp`]
Write the classes as presented in figures 10.10-10.13 of our Deitel textbook, with modifications per the universal requirements listed in lab 1a, and the grading policy. Name the files **Date.h**, **Date.cpp**, **Employee.h**, and **Employee.cpp**. Write your own test CPP(s) but do not submit them -- the instructor has his own test CPPs!

Name functions exactly as Deitel does.

Remember to not depend on dependencies. If a function name or class name or template name (etc) appears in an H or CPP file, #include and define it *IN THAT FILE*. Remember this requirement for ALL your work in COMSC 200.

Command Line Compiling With Visual C++:
Here is the command for compiling (e.g.):

```
cl test1.cpp Date.cpp -EHs
cl test2.cpp Employee.cpp Date.cpp -EHs
```

Alternatively, to compile each code module separately:

```
cl Date.cpp -c -EHs
cl Employee.cpp -c -EHs
cl test1.cpp -c -EHs
cl test2.cpp -c -EHs
cl test1.obj Date.obj
cl test2.obj Employee.obj Date.obj
```

Command Line Compiling With g++:
Here is the command for compiling in g++:

```
g++ test1.cpp Date.cpp
g++ test2.cpp Employee.cpp Date.cpp
```

These create executable named **a.out** on Macs and **a.exe** on PCs. To run on a Mac, type the command **./a.out**.

Alternatively, to compile each code module separately:

```
g++ -c Date.cpp -o Date.o
g++ -c Employee.cpp -o Employee.o
g++ -c test1.cpp -o test1.o
g++ -c test2.cpp -o test2.o
g++ test1.o Date.o
g++ test2.o Employee.o Date.o
```

*Post **Date.h**, **Date.cpp**, **Employee.h**, and **Employee.cpp** to the COMSC server for credit.*

Lab 5b Writing `friend` Functions [`ShortestRoute1.cpp`]
A classic problem in mathematics is the linear programming problem of finding the shortest route through a network of interconnected points. Applications include finding the shortest distance from one city to another, and finding the critical path in a project.

The solution is complicated. It requires three classes: `Leg` (to be developed in this lab), `Route` (to be developed in labs 6 and 7), and `ShortestRoute` (to be developed in the final project). The purpose of `Leg` is to represent the fact that two points in a network are connected, and to store the distance between the two. The purpose of the `Route` class is to link adjacent `Leg`s to form a route connecting any two points in the network. The `ShortestRoute` class contains an algorithm to find the shortest route between any two points.

We will not complete the full solution until the final project. But we will start by building and testing the `Leg` class.

For this assignment, write a program file named **ShortestRoute1.cpp**, to design and test a class that represents a leg of a route between two adjacent cities. Here are the specifications for the `Leg` class:

1. Include the following three *private data members*: the starting city, the ending city, and the distance (in miles) between the two. Use either `const char* const`'s or `const char` arrays to store the city names, and any constant numeric data type to store the distance.

2. Include one **`public`** *constructor*. Its parameter list should include a value for setting each private member variable. Initialize each of the member variables in the constructor to these parameter values. Do *not* include a default constructor in the `Leg` class, either now or in future versions.

3. Include a void *friend* **`outputLeg`** function, to output a textual description of the leg, including understandable labeling and summarizing all of its member variables. (For example, "San Francisco to San Jose, 20 miles"). The prototype should be as follows: `void outputLeg(ostream&, const Leg&)`, where the first parameter is to be used to pass `cout` -- name it `out` in the function. By including the **`outputLeg`** function as a friend of your class, you can send it to `cout` as follows:

```
Leg a("San Francisco", "San Jose", 20.0);
outputLeg(cout, a);
```

Note that the output function cannot be a member function. It could be a stand-alone, but that would require getters, and we have no getters. The specification is for a friend function for the purpose of

learning about friend functions.

Also note that this is the first time we've run into `ostream`. The correct way to get this is from the `iostream` library. Be careful, because you'll see other ways to include this, and they may work for the compiler being used, but iostream is the correct libary, and works with all compilers I know about.

4. Include a **`main`** function to test the class. In **`main`**, instantiate at least 5 objects of the Leg class -- you may use an array, if you wish. Name the cities and set their distances as you wish, so that when put together, they represent a route -- so the city name at the end of a leg must be the same as the starting city of the next leg. Output each object -- if you use an array to store the objects, you may use a `for` loop to output them.

5. Create ALL Legs first, then output them all.

Do *not* use more than one source file -- that means no **.h** files, and just a single **.cpp**. *Post **ShortestRoute1.cpp** to the COMSC server for credit.* Save this file, because modifications will be made to this program in labs 6 and 7, and the final project. The ShortestRoute series is the subject of three (3) lab assignments, and the final project.

How to pause a console program: *show / hide*

GRADING POLICY *show/hide*
