MCIS-4130
Homework Assignment 4

Purpose:

This assignment provides opportunities to work with references, function overloading, C++ extensions to `struct`s, and to gain more experience in modeling.

What to hand in:

Hand in the test program for part A, together with a header file for the Date `struct` in the standard form and one or more implementation files. Also hand in the output of sample runs of your program (or a listing of error messages if it didn't run).

Hand in the test program for part B, together with a header file with the *print()* declarations requested (and no others!) in the standard form, and one or more implementation files. Include answers to the questions. Also hand in the output of the program (or a listing of error messages if it didn't run).

Note: This assignment requires two separate main programs: However, you are permitted (and encouraged) to use the date header and implementation from part A in part B. That is, you will create, say, date.h, date.cpp, and testdate.cpp for part A; and print.h, print.cpp, and testprint.cpp for part B. You should use date.h and date.cpp from part A to perform the date printing required in part B.

Problems:

A. Improve your *Date*

Reimplement your *Date* structure by putting the functions inside of the structure declaration (you will make this a full-fledged class *next* week). This makes the functions callable with the "dot" notation. The function definition may reference the structure components by name for the current instance (e.g., a *Print()* function inside the *Date* structure could reference the *day* component as if it were the variable *day*). Be sure to include the scope operator in defining the function. Change the names of the functions to reflect their new scoping. Update your test program to test the new approach. Implement and test at least the following functions

Get the day (from instance)	*Get the month (")*	*Get the year (")*
Create (instance from data)	*Create (instance based on current date)*	
Add/subtract days	*IsValid (current instance)*	

- Optional: Create a new *Print()* function or change the *Print()* function above to use *ostream&* instead of defaulting to *cout*. Create the overloaded output operator
  ```
      ostream& operator << (ostream& os, Date& d)
  ```
  implemented with the *Print()* function mentioned above.
- Optional: Consider *Days between two dates*
- Optional: *Read a date (any istream)*

B. An overloaded print function

Write a set of overloaded global functions called *print()* which can print *int*, *signed char*, *float*, *double*, *char* arrays, and *Date* variables (***and no others***). Be sure to build an appropriate header file so the function prototypes are available.

Write and hand in a program that tests these functions.

Include answers for the following questions (these are for credit):

Q1. What happens when you try to *print()* an *unsigned long*? Why?

Q2. What happens when you *print()* an *unsigned char*? Why?

Q3. What happens when you call *print('a'+1)*? Why?

Notes:

For Part A, please use the "mechanical" transformation techniques we discussed in class to move from the procedural implementation in C to the abstract data type implementation in C++. Afterward you may add optional portions, or rename existing functions.

For Part B, please ensure you have test cases for all functions you provided, and test cases for answering the questions.

Questions posed here are for thought, not for written responses. We will discuss these next week.

How can we solve the problems posed in part B above?

Why *must* one use C++ to perform parts A and B of the problem?

Consider changing parameter passing mechanisms in parts A and B. What happens if you change from passing a value to passing a pointer? What happens if you change from passing a value to passing a reference? When shouldn't you use a reference?

Evaluation criteria:

In parts A and B, program correctness and completeness should be emphasized; be thorough with your testing. Good style should be second nature by now.

Note: DO NOT combine the main programs for the two exercises.

15	Part A: Header File
15	Part A: Implementation files
20	Part A: Test files and output ***Note: If your program doesn't compile, hand in the error listing as the results!***
15	Part B: Header File
15	Part B: Implementation files
20	Part B: Test files and output ***Note: If your program doesn't compile, hand in the error listing as the results!***