CPSC 131, Data Structures – Fall 2021
Project 1: Introduction & Review

Learning Goals:

- Become familiar with creating, compiling, running, and submitting programming assignments
- Demonstrate mastery of basic C++ skills, including
  - allocating and releasing dynamic memory
  - reading from standard input and writing to standard output
  - overloading insertion and extraction operators
- Demonstrate the ability to translate requirements into solutions
- Refresh your memory and normalize our point of departure. Depending on your background and how long ago you actively practiced programming in C++, some of this may be a review and some may seem new to you.

Description:

In this assignment, you will implement a class that obeys object-oriented programming principles. The class Book represents a book that could be sold by a retailer such as Amazon or Barnes & Noble. The class has a few private attributes, and a multiparameter constructor. Objects of the class have the fundamental capability to insert, extract, and compare themselves. You may reuse class Book in future homework programming assignments, so the effort you apply now getting it right will greatly benefit you throughout the entire semester.

What to Do:

1. You are to modify only designated TO-DO sections. The grading process will detect and discard any changes made outside the designated TO-DO sections, including spacing and formatting. Designated TO-DO sections are identified with the following comments:

```
//////////////////////// TO-DO (X) ////////////////////////////////
...
//////////////////////// END-TO-DO (X) ////////////////////////////
```

   Insert your code between the comments. When you have completed a TO-DO, delete the TO-DO and END-TO-DO comments. All of the TO-DOs are in Book.cpp.

   Hint: In most cases, the requested implementation requires only a single line or two of code. Of course, finding those lines is non-trivial. All can be implemented with less than 7 or 8 lines of code. If you are writing significantly more than that, you may have gone astray.

   Additionally, you will be writing main.cpp, such that when it is provided expected_input.txt as input, it provides output similar to that found in expected_output.txt.

Reminders:

- The C++ using directive `using namespace std;` is **never allowed** in any header or source file in any deliverable product. Being new to C++, you may have used this in the past. If you haven't done so already, it's now time to shed this crutch and fully decorate your identifiers.
- It is far better to deliver a marginally incomplete product that compiles error and warning free than to deliver a lot of work that does not compile. The autograder rejects submissions that do not compile. A rejected submission receives an "F" score (approximately 20%). It doesn't matter how pretty the vase was, if it's broken nobody will buy it.

- Object Oriented programming suggests that objects know how to read and write themselves. Classes you write shall overload the insertion and extraction operators.
- Object Oriented programming suggests that objects know how to compare themselves. Classes you write shall overload the equality and inequality relational operators.
- Always initialize your class's attributes, either with member initialization, within the constructor's initialization list, or both. Avoid assigning initial values within the body of constructors.

Team Formation:

As stated in the syllabus, you may work in a team of 1-2 students. Your first task is to decide on who will be on your team before you do any of the work described below.

In particular, you need to communicate with your team members, and reach consensus on the members of the team, before accepting the assignment invitation in Github. Teams cannot be changed in Github, so once you form a team there, you are stuck with it for the remainder of this project. (You can form different teams on other assignments.)

Obtaining and Pushing Code:

We are using:

- GitHub Education to distribute starter code. You save your work by pushing to a github repository;
- Ubuntu-latest within Github actions to run and evaluate your code
- Canvas is where your grade will be reported

We are using GitHub Education to distribute starter code; Tuffix to run and grade code; and GradeScope to collect submissions.

This document explains how to obtain starter code and push it to GitHub: GitHub Education / Tuffix Instructions.

This Youtube video by David McLaren explains how to get started with GitHub: https://youtu.be/1a5L_xsGIm8.

Here is the invitation link:

https://classroom.github.com/a/1yzCQ0Yx

Code Layout:

You will need to work with the following files:

- `README.md`: You must edit this file to include your name and CSUF email. This information will be used in the grading process.
- `Book.hpp`: This file declares the Book class, and is complete. Do not modify this file.
- Book.cpp: Definitions for all the Book member functions. All of your code changes go in this file, between the TO-DO comments.
- `Book_test.cpp`: The main function tests the output of your functions. Do not modify this file.

The repository also contains the following files, which you must leave unchanged, and may ignore. (You're welcome to look inside if you wish.):

- .gitignore configures git to ignore temporary files.
- CMakeLists.txt: Configures the automated building process that is utilized by Github Actions, centralizes differences in projects allowing for a unified compilation command. Do not modify this file.
- local_build.sh: Runs a sequence of commands that accomplishes the same outcome in the local terminal as what is produced using github actions. A useful way to test without access to GitHub.
- LICENSE assigns the MIT License to the starter code.
- c-cpp.yml - Configures and contains the commands issued by Github Actions when certain actions (such as pushing/pulling) occur. Do not modify this file.

Grading and Deadline

The project deadline is due **Tuesday September 27, 11:59pm**, which includes the 24 hour automatic extension. Submissions are not accepted by any means after the deadline has passed.

As stated above, the code assignment will be graded automatically by Gradescope using `make grade`.

The rubric includes points for **Code Design and Style**. Your instructor will read your code and evaluate

- completeness (code works in general, not just in unit test cases)
- code design (e.g. encapsulation, reuse, proper use of helper functions and constants)
- code style (consistent whitespace and formatting; appropriate variable names and comments)
- C++ practices (proper ownership/memory management, pass-by-reference, avoiding gross inefficiency)

Be advised that we use automated tools to detect plagiarism. Do not plagiarize. As stated in the syllabus, a submission that involves academic dishonesty will receive a 0% score (on both the code and report) and the incident will be reported to the Office of Student Conduct. A repeat offense will result in an "F" in the class and will be reported to the Student Conduct office.