Due Date: Feb 21, 2020 @ 11:59:59

STL Introduction – BookDB

This assignment will focus on using a few features of the C++ Standard Library. You will write a program that uses a vector to construct and manipulate a simple database of books. Each book instance will be represented as a struct, and you will be required to traverse, modify, sort, and delete from the database.

   You will submit your work to Gradescope and it will be both autograded and human graded. The autograder will compile your code with g++-7.4.0. Be sure to watch Piazza for any clarifications or updates to the assignment.

> Don't forget the Assignment 2 Canvas Quiz!
> Your assignment submission is not complete without the quiz.

BookDB Overview

Your book database must consist of 3 files: `BookDB.h`, `BookDB.cpp`, and `demo.cpp`. The first being the header file, the second being the implementation of the database modification functions, and the third being the `main()` function implementation.

For `BookDB.h`:

- Be sure to have include guards.

- Don't import anything that is unnecessary.

- Include file header comments that have your name, student id, a brief description of your book database program, and be sure to cite any resource (site or person) that you used in implementing this assignment.

- Each function should have appropriate function header comments.

    - You can use javadoc style (see doxygen) or another style, but be consistent.
    - Be complete: good description of the function, parameters, and return values.

- As in assignment 1, no classes and no inline functions.

- Each Book has the following features associated with it:

    1. Book ID (int)
    2. Publication Year (int)
    3. Rating (double)

For `BookDB.cpp`:

- All the function bodies should be here (except `main()`).

- Don't clutter your code with useless inline comments, unless it is the "why".

- Follow normal programming practices for spacing, naming, etc: be reasonable and consistent.

- Be sure to avoid redundant code.

For `demo.cpp`:

- The implementation of `main()` should be here.

- Follow the output format specified below, call the functions you implemented in `BookDB.cpp`, and print any necessary text to `std::cout`.

- Refer to the sample output linked below to compare your output to the expected output.

BookDB Valid Commands

Command Format	Description
`p`	Print the current contents of the database: for each book in the database, print its ID, publication year, and rating (separated by commas) on a new line. If no entries are in the DB, print "No entries" to `cout`.
`a <ID> <year> <rating>`	Add a book to the database with the given fields. Print "Book <ID> added" if successful.
`d <ID>`	Delete the book with the given ID. Print "Book <ID> removed" if successful.
`u <ID> <year> <rating>`	Locate the book with the given ID and update the value of its year and rating. Print "Book <ID> updated" if successful.
`c`	Calculate and print the average rating of all the books in the database in the format "Average rating = <rating>".
`f <year>`	Find and print all books in the database that were published in the given year in the format specified by the p command above. If no entries are found, print "No entries" to `cout`.
`s <sort option>`	Sort your book database in different ways so that print() displays the books in different orders. The valid sort options are 1, 2, or 3. The sort behavior is as follows: • 1: Sort the books in order of increasing ID number. • 2: Sort the books in order of increasing year. Break ties by putting smaller book IDs before larger ones. • 3: Sort the books in order of increasing rating. Break ties by putting smaller book IDs before larger ones.
`q`	Quit the application.

Functions to implement:

`int main()` Your main function will repeatedly prompt the user to choose from a set of operations, process the database accordingly, and display the resulting output on the console window. Keep prompting for a user command using a single ">" character on a new line until the letter "q" is entered, at which point your program should output "quit" and exit with a return code of 0. Anytime one of your other functions returns 1 (or -1.0 for `calculateAverageRating()`), this function should print "Error processing command" to `cout`.

`int addBook(int bookID, int year, double rating, std::vector<Book> &db)` First search the database for an entry with the given bookID. If no such entry is found, add a book with the

given bookID, year, and rating to the end of the vector and return 0. If there is already an entry with the given bookID, the list should not be changed and 1 should be returned.

`int updateBook(int bookID, int year, double rating, std::vector<Book> &db)` Search the database for an entry with the given bookID. If such an entry is found, update the rating and the year as per the input and return 0. If an entry is not found, return 1.

`int deleteBook(int bookID, std::vector<Book> &db)` Search the database for an entry with the given bookID. If such an entry is found, remove the entry from the database and return 0. If there is no entry on the list with the given bookID, the list should not be changed and 1 should be returned.

`std::vector<Book>* findBooks(int year, const std::vector<Book> &db)` Returns a pointer to the vector of Books published in the year specified by the argument. Don't forget to allocate and free heap memory for the vector!

`double calculateAverageRating(const std::vector<Book> &db)` Calculate and print the average rating of all the books in the database. If no books have been added, return -1.0 and process it as an error in `main()`.

`void print(const std::vector<Book> &db)` Prints the contents of the database in the format specified in the description above to `cout`. **DO NOT** check for an empty list here– check first **before** calling this function.

`int sortDB(std::vector<Book> &db, int sortingMethod)` This functions should take two arguments: the book database and the user-specified sorting behavior choice. After calling this function, a call to `print()` from `main()` should produce a sorted output. If the user enters something other than the three possible numbers for the sort option, return 1. Otherwise return 0.

Notes:

- The only functions that should be printing to `cout` are `main()` and `print()`.

- End each line of output with `std::endl`.

- You are free to create more functions than the ones listed, just make sure that the function signatures and names in your BookDB.cpp file match the ones specified in this write-up.

- You can assume that we will test with the correct type and number of arguments for each of the commands listed in the table above.

- Sample input/output can be found here.

Makefile

Your submission must include a `Makefile`. The minimal requirements are:

- Compile using the demo executable with the `make` command.

- Use appropriate variables for the compiler and compiler options.

- It must contain a clean rule.

- The demo executable must be named something meaningful (not a.out).

README.md

Your submission must include a `README.md` file (see https://guides.github.com/features/mastering-markdown/). The file should give a brief description of the program, the organization of the code, how to compile the code, and how to run the program.

Submission

You will upload all the required files to Gradescope. Again, don't forget to complete the Assignment 2 Canvas quiz.

There are no limits on the number of submissions. See the syllabus for the details about late submissions.

Grading

For this assignment, half of the marks will come from the autograder. For this assignment, none of the test details have been hidden, but there may be hidden tests in future assignments.

The other half of the marks will come from human-grading of the submission. Your files will be checked for style and to ensure that they meet the requirements described above. In addition, all submitted files should contain a header (comments or otherwise) that, at a minimum, give a brief description of the file, your name, and wisc id.

HAPPY CODING!