CS 474: Object Oriented Programming Languages and Environments

Spring 2013

First C++ project

Due time: 7:00 pm on Friday 4/5/2013

You are required to implement a *Painting manager* system in C++. This system keeps track of paintings in an art gallery. Each painting conforms to the structure below. For this project, paintings have a unique format:

1. *Title*—This is the title of the painting; it does not need to be unique with respect to other paintings.
2. *Artist first name*—The first name of the artist who painted the painting.
3. *Artist last name*—The first name of the artist who painted the painting.
4. *Height*—The height of the painting.
5. *Width*—The width of the painting.

Your painting manager must organize its paintings as a collection of linked lists. Each linked list will hold paintings of a given artist (as defined by the artist's first and last name). You must define at least three C++ classes, namely *LinkedList*, *Painting* and *String* (for the paintings' titles, artists' first and last names and so on). The classes are not related by inheritance. Each class will be appropriately equipped with constructors and destructors. The constructors must include at least a programmer-defined default constructor, and a copy constructor performing a deep copy of the receiver. In addition, you must support the functionality below. In what follows, you may assume that the title of each painting is unique; you can use the title of the painting as a unique identifier for the painting. Also, the painting's title will not contain any blank characters. Use a command line interface for entering the commands below. Your command line interface will prompt the user for a command, and then execute the command. Here is a list of commands. Make sure not to cause any memory leaks or dangling pointers in the implementation of these commands.

1. `a`—Add a painting to the collection kept by your manager. The user is prompted for the items of information pertaining to the painting and then an object corresponding to the painting is added to an appropriate list for the painting's artist. If this is the first painting of this artist, a new linked list corresponding to this artist is created and an object corresponding to the painting is added to the list.
2. `r`—Removes a painting from the collection. The user is prompted for the painting's title and then the painting is removed from the linked list corresponding to the painting's author. If a painting witht the given title does not exist, a message is displayed on the standard output device, and no further action is taken. However, if the painting exists and if this was the only painting of a given artist, the linked list corresponding to the artist is deleted altogether.
3. `d`—Deletes all paintings by a given artist. The linked list containing the all the painting of a given artists is removed from the painting manager.
4. `l`—Lists all the paintings. All painting in the paintings manager are printed by artist's name. The list does not need to be sorted by paintings title or artist's name.
5. `c`—Copy a painting. A painting with a given title is copied and the copy is added to the collection kept by your painting manager. The title should varied (e.g., by appending a string such as "version 2") with respect to the original painting. Otherwise, the copy and original painting have the same content. The copy and original painting are not allowed to share any data structures (i.e., you must perform a "deep copy").
6. `q`—Quits the painting manager.

You must work alone on this project. Save all your code in a collection of header and code files and submit a zip archive with a (short) readme file containing instructions on how to use your Painting Manager. Submit the archive by clicking on the link provided with this assignment. Your code should compile under the GNU C++ compiler. No late submissions will be accepted.