A CD Collection Catalogue

Due: 23:59 Wednesday, January 20, 2016

NOTE: You are expected to be familiar with the submission requirements and code quality guidelines outlined at the course webpage. Failure to adhere to these guidelines will result in mark deductions.

In this assignment you are to implement a `CDCatalogue` class for your legally purchased music collection that stores a set of `CD` objects. The catalogue should use a dynamic array for its underlying data storage requirements.

Please read the requirements carefully, paying particular attention to the names and input/output requirements of the class and its methods. As mentioned on the course webpage, your code will be tested using a prepared test suite – your code must compile in order to receive any credit, so at the very least you must write stubs for any functions which you are unable to complete. You will be supplied with a partial test suite which you can follow as an example for testing your functionality. The test suite which we use to grade your submission will be much more complete, so it is up to you to ensure that your `CDCatalogue` class is thoroughly tested with all general + special cases.

`CD` class

You are provided with a complete `CD` class with which to use for this assignment. Please download cd.h and cd.cpp from the course webpage. No modifications will be necessary to these files; methods are documented in the `CD` class header file.

`CDCatalogue` class

You are provided at the course webpage with a fully documented cdcatalogue.h header file which will not need to be modified. You are to complete the implementation of the `CDCatalogue` class in a cdcatalogue.cpp file.

Notes

String Objects

The `CD` class makes extensive use of string objects. You can find information about the standard string class at: http://www.cplusplus.com/reference/string/string/ . The main things you need to know about strings for this assignment are:

- You can assign an existing string variable or a string literal to a string variable
  - e.g. `string str1 = "bob";` or `str1 = str2;` (where `str1` and `str2` are strings)
- You can use the normal comparison operators (`<`, `>`, `<=`, `>=`, `==`) to compare strings
  - You do not need to know anything about the underlying representation of strings – the operations described above are all you need to do with strings in this assignment.

Set Operations

The `Join`, `Common`, and `Split` functions in `CDCatalogue` are analogous to the set union, intersection, and difference operations which should be familiar from MACM 101. Recall that:

- A set cannot contain duplicate values
- The union of two sets R ∪ S is the set of values that appear in either R or S (or in both)
  - e.g. $\{1,3,4,5\} \cup \{2,3,4\} = \{1,2,3,4,5\}$
- The intersection of two sets R ∩ S is the set of values that are common to both R and S
  - e.g. $\{1,3,4,5\} \cap \{2,3,4\} = \{3,4\}$
- The set difference of two sets R − S is the set of values that appear in R but not in S
  - e.g. $\{1,3,4,5\} - \{2,3,4\} = \{1,5\}$

The precise details of duplicate qualification for the purposes of the `Join`, `Common`, and `Split` functions are documented in cdcatalogue.h.

Function Stubs

If you are unable to complete the functionality of a function, you should still complete a stub so that the function compiles, although it will not return the correct value. An example stub for the Remove method:

```
// Stub for Remove method
bool CDCatalogue::Remove(CD disc)
{
  return false;
}
```

Deliverables

Your submission should include the following files:

- A title page listing the name(s) and information of contributing group member(s)
- cdcatalogue.cpp

Only one submission is required for a group. Include the above deliverables in a zip archive titled "assign-1.zip" and upload it to the CourSys submission server.