Library (Dynamic Array-Based Unsorted List)

Objectives

- Be familiar with object-oriented programming concepts.
- Be able to implement unsorted lists using statics array.

Instructions

In this programming assignment, you are about to write an object-oriented program to mimic a library. Each book in the library is represented by a unique ISBN number. Librarians are able to search, add, and delete books from library catalogs. Library patrons are able to search and print out the information of a book (in this case only ISBN). The following are the tasks you will complete:

1. Use the program templates in the shared folder
   `\\copernicus\public\csc2320\programmingAssignments\prog1`
   to complete this assignment. You are NOT allowed to change the header files and client.cpp.

2. Create a C++ class named Library and a class named Book. You need to implement Class Library via a dynamic array so that the capacity of the library (that is, the maximum number of books) can be specified by the users. In doing so, in class Library define two private member variables:

   Book * books;
   int numOfBooks;

   Variable "books" is a pointer to a Book object. In the constructors of class Library, you need to use "new" operator to create a dynamic array and let variable "books" point to the array. Also, you need to have a user-defined destructor for class Library to release the memory of the array dynamically created, because it is placed on the heap memory region and needs to be removed by programmer once the function where the array variable resides is returned.

3. Class Book includes a private member variable **isbn** and constructors and a number of public functions.

4. The public functions defined in class Library include a parameterized constructor, a copy constructor, a destructor, search, insert, delete, and print functions, etc.

5. In the client program, there are two Library objects being created. Observe the number of books in each Library object and make sure your program outputs correct results.

6. Thoroughly test your program to make sure all possible cases are tested. A sample test is given in the bottom section of client.cpp as comments.

7. Upon completion, zip all your source code and test output into a single zip file and then submit to the Blackboard. Please don't submit your entire project folder. Our TA will run your program using client.cpp in his project folder.