Exercise 2: Simple Library Management System

Introduction

This exercise is designed to help you review the basic concepts and skills in C++ programming. The focus of this exercise is mainly on `struct` operations. Don't worry, they're just meant to be a warm-up for the upcoming exercises and projects.

Overview

In this exercise, you will implement a simple library management system. It manages book collections with functionalities to **add, borrow, return and view the library's current inventory**. Through this exercise, you will practice your skills in struct usage, array management, and basic C++ operations.

Task

Structs and Constants Setup

To begin with, you need to define a struct `Book` to represent a single book. The struct should have the following elements:

- `ID` : an `int` to represent the book's unique identifier. It is used to track and manage books within the library. The ID of the first book is 1.
- `title` : a `string` to represent the book's title.
- `author` : a `string` to represent the book's author.
- `isAvailable` : a `bool` to represent whether the book is available for borrowing. It is `true` if the book is available, and `false` if the book is borrowed.

Also, you need to define a constant `MAX_BOOKS` to represent the maximum number of books that the library can hold. In this exercise, set `MAX_BOOKS` to 10.

To represent the entire library system, you need to define a struct `Library` to hold the book collection. The struct should have the following elements:

- `books` : an fixed-size array of `Book` to represent the library's book collection. The size of the array is specified by `MAX_BOOKS` .

- `numBooks`: an `int` to represent the number of books currently in the library.

Library Management Functions

The functions you need to implement are as follows:

`Library initLibrary()`

- **Effects**: Returns a `Library` with no books (i.e., `numBooks` is 0).

`void addBook(Library &lib, std::string title, std::string author)`

- **Requires**: `lib` is not full.
- **Modifies**: `lib`.
- **Effects**: Adds a book with the given title and author to `lib`. Each new book is assigned an ID, which is its index in the `books` array plus one. If `lib` is full, prints "The library is full." followed by a newline.

`void borrowBook(Library &lib, int ID)`

- **Requires**: `ID` is a valid book ID. If the input `ID` is not valid, print "Invalid book ID." followed by a newline.
- **Modifies**: `lib`.
- **Effects**: Sets the availability of the book with the given ID to `false` if the book is available and prints "Book `<title>` is borrowed." followed by a newline; otherwise, prints "Book `<title>` is not available." followed by a newline.
- **Sample Output**: `Book The Catcher in the Rye is borrowed.`

`void returnBook(Library &lib, int ID)`

- **Requires**: `ID` is a valid book ID. If the input `ID` is not valid, print "Invalid book ID." followed by a newline.
- **Modifies**: `lib`.
- **Effects**: Sets the availability of the book with the given ID to `true` if the book is not available and prints "Book `<title>` is returned." followed by a newline; otherwise, prints "Book `<title>` is already available." followed by a newline.
- **Sample Output**: `Book 1984 is already available.`

`void printLibrary(const Library &lib)`

- **Effects**: Prints all books in `lib` with their IDs, titles, authors, and availability followed by a newline; if `lib` is empty, prints "The library is empty." followed by a newline.
- **Sample Output**: See the testing section.

Implementation Details

- You should declare all the structs, constants and functions in `ex2.h` and implement the functions in `ex2.cpp`. **You need to write all the files from scratch.**
- You **should not** use dynamic memory allocation or any STL containers for the `books` array in the `Library` struct. It should be a fixed-size array.
- Note that the book `ID` starts from **1**, while the index of the `books` array starts from **0**. You need to handle the conversion between the two.
- You may name the struct members as you like, **but the function names and the order of the parameters should be consistent** with the function descriptions.
- You may assume that no books will be removed from the library. And each book added to the library will have a unique title.
- Handle exceptions and prints error messages as specified in the function descriptions. **Any typo or format error will be considered as failed test cases.**

Testing

You may write your own `main.cpp` as the driver program and test your implementation. A sample `main.cpp` is as follows:

```
#include "ex2.h"
int main() {
  Library lib = initLibrary();
  addBook(lib, "The Catcher in the Rye", "J.D. Salinger");
  addBook(lib, "To Kill a Mockingbird", "Harper Lee");
  printLibrary(lib);
  borrowBook(lib, 1);
  borrowBook(lib, 2);
  returnBook(lib, 1);
  returnBook(lib, 1);
  printLibrary(lib);
  return 0;
}
```

Compile and run the program, and you may see the following output:

```
Book ID: 1
Title: The Catcher in the Rye
Author: J.D. Salinger
Status: available
Book ID: 2
Title: To Kill a Mockingbird
Author: Harper Lee
Status: available
Book The Catcher in the Rye is borrowed.
Book To Kill a Mockingbird is borrowed.
Book The Catcher in the Rye is returned.
Book The Catcher in the Rye is already available.
Book ID: 1
Title: The Catcher in the Rye
Author: J.D. Salinger
Status: available
Book ID: 2
Title: To Kill a Mockingbird
Author: Harper Lee
Status: not available
```

Exercise 2 will be tested on JOJ and will have hidden test cases. Make sure your implementation is correct and efficient.

If you find anything wrong or ambiguous in the description, feel free to post on piazza or contact us anytime.

Submission

Zip your `ex2.h` and `ex2.cpp` and submit them to JOJ. **The due date is Oct. 22nd.**