

You will be writing a Library simulator. It will have three classes: Book, Patron and Library. To make things a little simpler for you, I am supplying you with the three .hpp files. You will write the three implementation files. You should not alter the provided .hpp files.

A couple of notes:

- The `vector::erase()` function lets you delete an element of a vector, shifting over all the elements after it.
- You'll see in `Book.hpp` a line that says `"class Patron;"`. That is a forward declaration. It doesn't say anything about the definition of the Patron class, but it promises the compiler that there will be a type named Patron. The reason we don't just say `"#include Patron.hpp"` is that both Book and Patron need to know about each other, but they can't both `#include` the other because that would create a cyclic dependency.

Here are the .hpp files: [Book.hpp](#), [Patron.hpp](#) and [Library.hpp](#)

Here are descriptions of the three classes:

Book:

- `idCode` - a unique identifier for a Book (think library bar code, not ISBN)
- `title` - cannot be assumed to be unique
- `author` - the `idCode`, `title` and `author` don't need set methods, since they will never change after the object has been created, therefore these fields can be initialized directly within the constructor
- `location` - a Book can be either on the shelf, on the hold shelf, or checked out
- `checkedOutBy` - pointer to the Patron who has it checked out (if any)
- `requestedBy` - pointer to the Patron who has requested it (if any); a Book can only be requested by one Patron at a time
- `dateCheckedOut` - when a book is checked out, this will be set to the `currentDate` of the Library
- `CHECK_OUT_LENGTH` - constant that gives how long a Book can be checked out for
- constructor - takes an `idCode`, `title` and `author`; `checkedOutBy` and `requestedBy` should be initialized to `NULL`; a new Book should be on the shelf
- some get and set methods

Patron:

- `idNum` - a unique identifier for a Patron
- `name` - cannot be assumed to be unique

- checkedOutBooks - a vector of pointers to Books that Patron currently has checkedOut
- fineAmount - how much the Patron owes the Library in late fines (measured in dollars); this is allowed to go negative
- a constructor that takes an idNum and name and initializes fineAmount to zero
- some get and set methods
- addBook - adds the specified Book to checkedOutBooks
- removeBook - removes the specified Book from checkedOutBooks
- amendFine - a positive argument increases the fineAmount, a negative one decreases it

Library:

- holdings - a vector of pointers to Books in the Library
- members - a vector of pointers to Patrons in the Library
- currentDate - stores the current date represented as an integer number of "days" since the Library object was created
- a constructor that initializes the currentDate to zero
- addBook - adds the parameter to holdings
- addPatron - adds the parameter to members
- getBook - returns a pointer to the Book corresponding to the ID parameter, or NULL if no such Book is in the holdings
- getPatron - returns a pointer to the Patron corresponding to the ID parameter, or NULL if no such Patron is a member
- checkOutBook
 - if the specified Book is not in the Library, return "book not found"
 - if the specified Patron is not in the Library, return "patron not found"
 - if the specified Book is already checked out, return "book already checked out"
 - if the specified Book is on hold by another Patron, return "book on hold by other patron"
 - otherwise update the Book's checkedOutBy, dateCheckedOut and Location; if the Book was on hold for this Patron, update requestedBy; update the Patron's checkedOutBooks; return "check out successful"
- returnBook
 - if the specified Book is not in the Library, return "book not found"
 - if the Book is not checked out, return "book already in library"
 - update the Patron's checkedOutBooks; update the Book's location depending on whether another Patron has requested it; update the Book's checkedOutBy; return "return successful"

- requestBook
 - if the specified Book is not in the Library, return "book not found"
 - if the specified Patron is not in the Library, return "patron not found"
 - if the specified Book is already requested, return "book already on hold"
 - update the Book's requestedBy; if the Book is on the shelf, update its location to on hold; return "request successful"
- payFine
 - takes as a parameter the amount that is being paid (not the negative of that amount)
 - if the specified Patron is not in the Library, return "patron not found"
 - use amendFine to update the Patron's fine; return "payment successful"
- incrementCurrentDate
 - increment current date; increase each Patron's fines by 10 cents for each overdue Book they have checked out (using amendFine)
 - If a book is checked out on day 0, then on day 1, the patron has had it for 1 day. On day 21, the patron has had it for 21 days, so it is still not overdue. On day 22, the book is overdue and fines will be charged.
- **be careful - a Book can be on request without its location being the hold shelf**
(if another Patron has it checked out);

One **limited** example of how your classes might be used is:

```
Book b1("123", "War and Peace", "Tolstoy");
Book b2("234", "Moby Dick", "Melville");
Book b3("345", "Phantom Tollbooth", "Juster");
Patron p1("abc", "Felicity");
Patron p2("bcd", "Waldo");
Library lib;
lib.addBook(&b1);
lib.addBook(&b2);
lib.addBook(&b3);
lib.addPatron(&p1);
lib.addPatron(&p2);
lib.checkOutBook("bcd", "234");
for (int i=0; i<7; i++)
lib.incrementCurrentDate();
lib.checkOutBook("bcd", "123");
lib.checkOutBook("abc", "345");
for (int i=0; i<24; i++)
lib.incrementCurrentDate();
lib.payFine("bcd", 0.4);
double p1Fine = p1.getFineAmount();
```

```
double p2Fine = p2.getFineAmount();
```

This example obviously doesn't include all of the functions described above. You are responsible for testing **all** of the required functions to make sure they operate as specified.

You must submit on TEACH: **Book.cpp**, **Patron.cpp**, and **Library.cpp**. You do not need to submit the .hpp files.

In the main method you use for testing, you should only need to #include Library.hpp. Remember that your compile command needs to list all four .cpp files.

Just to think about: There are six possible changes in the location of a Book. Can all six occur?

There are a lot of details here - it's entirely possible I omitted something or mis-stated something. If you think that is the case, I encourage you to ask me about it.