Understanding

- I need to create a library simulator
  - It needs to have member functions for a book, library, and patron class
  - A menu file needs to be created, exposing the following functions
    - View patron info
    - View book info
    - Check out book
    - Return book
    - Place book on hold
    - Pay fine
    - Add a book
    - Add a member
    - Increment day
    - Exit

Design

(Provided book class with setters and getters implemented) - Easy enough, no PC required
(Provided patron class with setters and getters implemented) - Again, easy

---

Library.cpp

```
Constructor
{
        //reserve 100 slots for holdings and members
        //set current date to 0
}

void Addbook
{
        //get idcode, title, and author from user
        //check to see if the book exists (using getbook)
        //add the book to the holdings vector

}

void AddMember
{
        //get idCode and name from user
        //check to see if user exists (using get getpatron)
        //add patron to members vector
}

void CheckoutBook
{
        //checks to see if a book and patron exists
        //if they exist, get the location of the book
        //shelf —> check out
        //hold —> check user, if right user; check out
        //checked out —> print that it can’t be checked out
}

Get current date() //no explanation needed

GetPatron() //helper function that checks to see if a patron exists
GetBook() //helper function that checks to see if a book exists

IncrementDate()
{
        //adds one to the date
        //loops through users and adjusts fines as needed
}

PayFine()
{
        //adjusts fine by given double
}


RequestBook()
{
        //make sure patron and book exist
        //If book isn’t requested, then
                //if book on shelf, put on hold shelf
                //if book checked out, update requested by
        //if book is requested print error
}

ReturnBook()

{
        //check book checked out and exists
        //set to hold shelf or regular shelf if or if not requested
        //remove book from user
}

Viewbookinfo //just a bunch of cout boilerplate
Viewpatroninfo //same
```

Menu.cpp

This will just be a switch statement that accesses public library functions. I don't think any further explanation is needed at this point in the quarter.

Testing

Input (continue from row to row)	Output	Result
Use the add user to add three users (a, b and c)	No visible output, three users in vector	same
Use the add book to add three books (1 2 and 3)	No visible output three books in vector	same
Checkout books as follows: 1 -> a 2 -> b 3 -> c	No visible output, books are now checked out	same
View book infos	Shows books and who they are checked out to	same
View patron infos	Shows patrons, zero fines, and books they have	same
Request for patron b, book 1	No output	same
View book infos	Shows books and who they are checked out too. book 1 should show that it is also requested by patron b	same
Increment date to 22	All users should have 10 cents in fines	same
check in book 1	no output. Users will show correct books	same
User c tries to check out book 1	Cannot, as it is on hold for user b	same
User b checks out book 1	correct books shown for correct users	same
Pay fine of 10 cents for user b	user b should show 0 for fines and console should says o	same

Reflection

All in all, this was a fairly straightforward assignment. I almost think it would have been more fun if we had to implement the header files and figure out how to make our own makefile… but perhaps that is to be saved for another day. I don't think I really had any major issues in development, some of which can be attributed to having a good IDE (cLion by jetbrains, check it out!). I think that the hardest part was making sure that all of the requirements were met. I almost missed printing out the user's fine balance after they paid!

We covered a lot of ground in this class. I think that this project was a good one to end on, as it pretty much used every concept that we had to use in previous assignments, (plus enums… I guess). I'm looking forward to next quarter and getting into more advanced concepts. One thing that I think would have helped with this assignment is UNIT TESTS. Testing is getting really arduous, and I hope that we have the option to include automated tests next quarter.