Reflection

What Went Well

I think I structured my project well. Files are easy to find and the CMake files are easy to understand. I managed to understand CMake, code modularisation and the good practices that come with modularisation (eg. implementation hiding using `static` modifiers and hiding struct declarations) pretty well. I believe I understood the responsibilities between library user (library as in code library) and library programmer, which would be the interface and the backend in this case. Regarding that however, I would rewrite each function that creates and array and string to take in an initialised pointer with length as parameters, as it would make it more clear to a library user their responsibility to free the memory.

Hardest Part

I didn't find anything particularly hard. Tracking down segfaults was made trivial by using a step-through debugger and a watch. Combined with decent programming practices which reduced segfaults, developing this project was largely problem-free.

Testing

Function: `int store_books(struct FILE *file);`
Expected Behaviour:
- Returns 0
- `book_db` will be stored in file `file`

Checked Exceptions:
- Returns 1 if file is NULL (invalid file)

Function: `int load_books(struct FILE *file);`
Expected Behaviour:
- Returns 0
- `book_db` will contain all the books in file `file`

Checked Exceptions:
- Returns 1 if file is NULL (invalid file)

Function: `int add_book(struct Book book);`
Expected Behaviour:
- Returns 0
- `book_db` contains book `book`

Checked Exceptions:

- Returns 1 if book with id `book.id` already exists

Function: `int remove_book(unsigned int id);`
Expected Behaviour:
- Returns 0
- `book_db` no longer contains book with id `id`

Checked Exceptions:
- Returns 1 if `book_db` never contained a book with the specified id `id` in the first place

Function: `struct BookArray find_book_by_id (unsigned int book_ID);`
Expected Behaviour:
- Returns a BookArray with an array of length 1 with the book with id `book_ID`

Checked Exceptions:
- Returns a BookArray with a NULL pointer and length 0 if book wasn't found

Function: `struct BookArray find_book_by_title (const char *title);`
Expected Behaviour:
- Returns a BookArray of variable length that has books with specified title `title`

Checked Exceptions:
- Returns a BookArray with a NULL pointer and length 0 if book wasn't found

Function: `struct BookArray find_book_by_author (const char *author);`
Expected Behaviour:
- Returns a BookArray of variable length that has books with specified author `author`

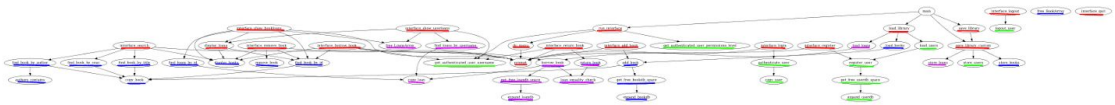
Checked Exceptions:
- Returns a BookArray with a NULL pointer and length 0 if book wasn't found

Function: `struct BookArray find_book_by_year (unsigned int year);`
Expected Behaviour:
- Returns a BookArray of variable length that has books with specified publication year `year`

Checked Exceptions:
- Returns a BookArray with a NULL pointer and length 0 if book wasn't found

Code Analysis (callgraphs using egypt)

Before:



The callgraph seems to have loosely coupled modules. The cluster of colours on the right is the load and save library functions, which call the load and save functions from all 3 of the library management modules. I believe that cluster can be safely dismissed as the rest looks mostly fine.

However, there is a glaring error on the top right, I never call free_BookArray(), which means when I call my search functions for book management I never free the struct. That was quickly fixed.