Dennis Krupitsky, Matthew Connelly, Andrew Teterycz

CSC 330 – Object Oriented Programming

Project #01 – Library Systems Information - Phase 3 (Test Pan)

Due Date: 04/17/2019

**Test Plan**

**Unit Testing** will be employed to test the Library System subsystems to make sure each subsystem behaves predictably on its own. Each “unit” will be a subsystem and therefore an object of a single class.

All constructors and methods will be called using a predetermined set of data as input. From this set of data, the ideal outcome of method calls and object construction can be modeled and tested against. If there are any inequalities or inconsistencies, they will either be caught by the testing system using basic Boolean expressions or by the developer, should the testing system fail to detect a bad outcome.

An ideal or desired object can be simulated using a struct with set initial values; this is a cheap and transparent way to simulate a desired object that the developer can comprehend at a glance in source code. This struct can act as a “mock class”.

For example: given the class BookInfo, the struct \_BookInfo would model the ideal BookInfo object. BookInfo getters and setters should behave the same as \_BookInfo member access and assignment, respectively.

int main()

{

BookInfo b2; // current object

cout << b1.title << endl;

cout << b2.getTitle() << endl;

cout << b1.title() == b2.getTitle() << endl;

}

Struct \_BookInfo

{

title = “Harry Potter”;

author = “J.K. Rowling”;

.

.

} b1; // desired object as struct

On the left is the definition of the \_BookInfo struct, and on the right is the driver where BookInfo’s title getter is called and \_BookInfo’s title member is accessed. The result of these two operations is then displayed as well as compared.