**Programming Assignment 3**

**Purpose**: Demonstrate the use of class inheritance, operator overloading, class dynamic data and error checking.

Program definition:

Begin with your solution to Assignment 2 and extend the solution in the following ways:

1. Derive an **extended** bookType to include the format of the book, i.e., paper back and/or hard back and the number of pages. This will be inheritance. DO NOT include the format of the book or number of pages in the original bookType definition. Create the format as a dynamic array. The datafile will now contain the number of different formats, the name of the formats, and the number of pages. For example,

Book Title

ISBN

Publisher

Year Published

Cost

Number In Stock

Number of Authors

List of Authors

Number of formats (up to 3)

**Format will be in the form:** (NOTE: may declare as an array. Three (3) points extra credit if format is declared as dynamic data)

**paperback**

**hardback**

**Number of pages**

At this point, you will replace the original object type from bookType to Extended bookType, i.e., you will no longer create booktype objects but objects of the inherited class.

The data read must be extended to read and set these values in the **extended** bookType data members.

1. In the inherited class, include the following:
   1. a data member to contain the formats available
   2. a data member to contain the number of pages
   3. a function to set the format and number of pages
   4. constructor(s) to initialize the number of pages and format
   5. override any output functions to include the number of pages and formats available
2. Include a process to “sell” books. This function will subtract the number of books entered at the prompt.
3. Add functions to:
   1. Overload the + operator (to add to the number of books)
   2. Overload the – operator (to subtract from the number of books)
   3. Consider the ways you need to add in overloading your arithmetic operators, i.e., book + number or number + book. Think about it…
   4. Overload the insertion operator for the class(es) in the print function. You will no longer call the print function. The output for your class information will look something like: cout << <classinfo> (where <classinfo> is the class object).
4. Include error checking as follows:

The number of books must **NOT** drop below 0 (zero), i.e., no negative numbers. If the ‘sell’ function will cause the number of books to drop below 0, display a message requiring the user to enter a number that will NOT cause this situation and loop until a correct value is entered.

All classes must have a class definition header file (\*.h) and a class implementation file (\*.cpp). You must have a main program that exercises these classes (\*.cpp). You must create a project in the IDE of your choice. You must submit all header files, source files (\*.cpp) and the project file (\*.dev; \*.cbp). Submit a README file that contains: The names of all your files including all the files in the project; how to compile; and how to run.

**All assignments are to be submitted to the appropriate Assignment posting in Canvas.**

**Submit your code files (\*.cpp, \*.h, \*.dev), and readme file**. Program **must** compile and all files must be submitted to receive credit. DO NOT submit the executable (\*.exe).

**A program that does not compile will receive a zero (0).**

You may zip all your files and submit the zip file. A zipped submission will receive 1 point extra credit.

**NOTE**: Since this assignment is an extension of assignment 2, I will restore points to assignment 2 if deficiencies identified in assignment 2 are corrected in assignment 3.

The code file **must** contain the following as documentation. If the following is not included, **10% may be deducted**:

The name of your C++ file

Your name

Some kind of date, either the due date or the date you finished

The type of input

The type of output

A brief description of the algorithm or purpose of the program

For example:

/ \* Program name: assignment1.cpp

Author: Pam Smith

Date: 8/25/18

Input: requests an input and output file name from the user (inputfile.txt provided).

Output is name of your choice

Output: displays output on the console and writes output to file name provided

Description: This program translates a word or phrase using the ROT13 cipher

\*/

**All** procedures and functions should be documented with the following information:

A brief description of the purpose of the function or procedure

Signature : return type, name, parameters (including type)

A list of the parameters and what each represents

A description of the function return type as applicable

Precondition(s) as applicable

Postcondition(s) as applicable

For example:

/\*

myfunction outputs the values contained in the input parameters.

myfunction returns no value; it takes an integer parameter as input and a character parameter as input

The integer parameter contains the ordinal number representing a letter in the alphabet

The character parameter contains a letter of the alphabet

No value is returned

Precondition: the integer value must contain a valid number representing the ordinal number of the letter contained in the character parameter; the character parameter must contain a value representing a letter of the alphabet

Postcondition: there are no post conditions

\*/

void myfunction (int a, char b) {}