Princess Sumaya University for Technology
جامعة الأميرة سمية للتكنولوجيا

King Hussein School for Computing Sciences
Department of Computing Science
CS11206 - Object Oriented Programming Assignment, Fall 2021/2022
Deadline: January 10, 2022, by 11:59pm

Objectives:

Practice the following:

1. Classes and composition
2. Dynamic Memory management for objects
3. Dynamic Memory management within class
4. this pointer
5. friend functions

Grading:

- **Your grade will mostly be determined based on a discussion with your instructor.**
- **You should read the "Coding Conventions" and "Academic Integrity" sections.**

	Max Mark	Member 1:	Member 2:	Member 3:
Question 1	55			
Question 2	35			
Coding Conventions	10			
Total	**100**			

Submitting your work

- **Deadline: January 10, 2022, by 11:59pm.**
- All your work must be submitted through eLearning portal
- Only one submission is required from each team
- Submit a compressed (e.g., a .zip) file that includes:
  - all your *.h header files and *.cpp source files
  - Include screenshots for the output of your program
  - A README text file with the following:
    - Names and IDs of all group members
    - A brief description of each team member contributions
    - What did you find most challenging about this assignment
    - Any other notes you would like us to know about your submission
    -

Question 1(55 pts): Implement the following classes:

A) [15 pts] **class Book** with the following members:

1. **private** members:
   a. book title (**string**)
   b. book ISBN (**string)**
   c. book first author (**string**)
   d. book number_of_copies (**int**)
   Note: for this assignment, a book ISBN can be any string (such as 978-3-16) and it should be unique for each book.
2. **public** members:
   a. A parameterized constructor with **default arguments** for all data members: '-' for the string data members and 0 for the numeric data member. The constructor should initialize all data members properly
   b. **setters** and **getters** for all data members. The setter for the **number of copies** should validate that the number is greater than or equal to zero. and for strings to be non-empty.
   c. The member function **updateBookNumCopies(int n)** to update the number of copies of a book. The function receives an integer value (can be negative or positive) and updates the current number of copies only If the result of adding that value to the current count is zero or more, otherwise, it prints a proper error message.

B) [40 pts] **class Library** where a library contains several books that can be **added or removed**. Implement class **Library** with the following:

1. Three private data members:
   a. **int maxSize** that represents the maximum number of books (i.e., **unique** book titles) that can be added to the library. Note that you can add any number of copies of a given book.
   b. **int current** that tracks the current number of unique book titles in the library. this attribute gets updated whenever a new book-title is removed or added to the library.
   c. **Book *bookList,** a pointer to a dynamic array of **Book**s that represent all unique book titles in the library.

2. The **private** member function **void resizeLibrary(int n)**, which resizes the dynamic array pointed to by the pointer **bookList.**
   a. The function should increase the array capacity by the amount of **n**.
   b. The function should validate that **n** is a positive integer.
   c. The function must preserve the data that was already in the list.

3. The following **public** member functions:

   Use ***this*** pointer in all member functions of the class that take one or more parameters.

   a. **Library(int n):** that sets **maxSize** to **n,** allocates dynamic space of size **maxSize,** and makes the **bookList** pointer points to the allocated space. and sets **current** to 0.
   b. A copy constructor **Library(const Library &)** to properly copy data from one Library object to another.
   c. **~Library():** Deallocates any dynamically allocated memory for an object.
   d. **isEmpty()**, a boolean function that returns true if the **library is empty**. and false otherwise.
   e. **isFull(),** a boolean function that returns true if the **library is full**, and false otherwise.

f. **int findBookByTitle(const string & title)** returns the index of the book with requested title if found in the library, otherwise returns -1. HINT: use a string compare function (strcmp) from the string library.
g. **int findBookByISBN(string isbn)** returns the index of the book with an ISBN that is equal to isbn if found in **bookList**, otherwise returns -1.
h. **void addBook(const Book & book)**: this function does the following:
   - adds the book to the end of list **bookList** if the **Library** is not full and the book *does not already exist* in the list.
   - If the book to is already in the list, then update the number of copies of that book (increment the current number of copies by the amount *number_of_copies*)
   - If a book is not in the list, but the list is full, then resize the library pointed. by **bookList** to allow for 10 more books then add the book at the first available location (i.e., after all the books that already exist).
i. **void addBook():** this function is similar to the previous method but this one asks for all required book data to be input by the user of your program.
j. Getters for **maxSize** and **current** data members.
k. **float capacityUsage**() that calculates the percentage of the maximum number of books filled with books. The division must be a floating-point division
l. **bool removeBookCopy(string isbn):**
   - if the book exists in the library then decrements the number of copies of the book by one
   - If the number of copies becomes zero, the function must delete the book from the list by shifting left all book items on the right of the book to be removed.
   - the function returns true if the book is found, otherwise, prints a proper message and returns false.

4. Define a global **friend** function **void printLibrary(const Library & libr)** that receives a Library object and prints the information of all the books in that library.

Note: You may implement additional methods if you need to.

Question 2 (35 pts): Write a driver program to test the classes you have implemented as follows:

Note: Do not reimplement functionalities that are already implemented in classes **Book** and **Library**.

1. Create a dynamic array of **Book** objects named **myBooks_items** of size **5**.
2. Fill **myBooks_items** with proper data from the user for 5 **different** books.
3. Create a **Library** object named **lib** with a **maxSize** of 6.
4. Fill the first 5 items of the library **lib** using the book items from **myBooks_items** array.
5. Create a dynamic object of type **Book** using the data of the second object in the array **myBooks_items**
6. Delete the dynamic array pointed to by **myBooks_items**
7. Add 3 more copies of the book from Question2-**point 5** to library **lib**
8. Add two more unique books by calling **addBook()**
9. display the details of all books in **lib**
10. call a proper function to print the **current** size of the library
11. Print the usage-percentage of the **bookList** in **lib** using a proper function call.
12. remove all copies of the book from Question2-**point 5**.
13. Once again, Display the details of all the books in **lib**.

Coding Conventions

1) Naming Conventions:
   - **Class** names should start with a capital letter. All other letters should be small except for the beginnings of words.
     Examples: BankAccount, GraduateStudent, RightTriangle.

   - **Local variable** names should be in small caps except for the beginning of words.
     Examples: numberOfCopies, bookTitle

     or words should be separated by underscores.
     Examples: number_of_copies, book_title

   - **Constants** should be named with all caps and words should be separated with underscores.
     Examples: PI, COMPANY_NAME

   - **Meaningful names** should always be used. Avoid names such as a, b, c, etc.

2) Comments:
   - **Each class** and function should be preceded by a comment that describes them. Add any other necessary important details.

   - **Data members** should be explained with comments preceding them only if not all information is clear directly from the name.

   - **Any line/part** of the code that is complex or is difficult to understand should be explained with appropriate comments preceding it.

   - Explaining **obvious** lines of code with comments should be **avoided**.

3) Interface/implementation
   - Any class interface should be separated from its implementation
   - A class interface should be in a header file (.h)
   - A class implementation should in a source file (.cpp)
   - your header files should be guarded against multiple inclusions using #ifndef, #define and #endif

4) Constant identifiers and constant member functions
   - Declare any class method that doesn't modify class data as a const member function
   - Use call by reference when it is appropriate to pass data to methods for efficiency
   - Use the const qualifier to protect data passed by reference unless the data needs to be modified.

Academic Integrity

1) **Collaboration Policy**: This course permits and encourages many forms of collaboration. However, you must be careful to collaborate using the following rules:
   - **No Visible Code:** When discussing ideas with other teams, you are not allowed to show your code (or pseudocode-like steps) and you are not allowed to look at code/pseudocode written by them.
   - You must acknowledge any collaboration you have had with anyone while working on the assignment.

2) **Outside sources.** Copying or adapting code that is not yours (e.g., from the internet) is NOT permitted.

3) **Penalties**. Penalties for plagiarism or abetting plagiarism (i.e., helping others plagiarize) can be any or all the following:
   - Zero on the assignment where plagiarism occurred.
   - A failing grade in the course.
   - A report written to the Deanship of King Hussein's College of Computing Sciences and the Deanship of Student Affairs.

ABET-specific note. This assignment addresses the following CLOs:

CLO1 Analyze user's programming requirements to identify their components
CLO2 Select appropriate IDE and download/configure it per host environment
CLO3 Design and Implement a suitable solution based on Object Oriented technology
CLO4 Evaluate and Contrast different design options in terms of effectiveness and efficiency
CLO5 Present solution to professor or committee to demonstrate effectiveness, highlight strengths vs. weaknesses and to justify technical choices