Department of Computer Science

COS110 2016
Practical 3

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

1 Introduction

1.1 Submission

The submission deadline for this practical is **5 September 2016 at midnight**. You should aim to complete this assignment before the due date.

Fitchfork marks by comparing the output of your program with specified expected output on a line by line basis. For this reason you should pay close attention to the instructions for the output of your program. Also remember that names of files are case sensitive.

1.2 A Serious Warning

It is in your own interest that you, at all times, act responsible and ethically. As with any work done for the purpose of your university degree, remember that the University of Pretoria will not tolerate plagiarism. Do not copy a friend's assignment or allow a friend to copy yours. Doing so constitutes plagiarism, and apart from not gaining the experience intended, you may face disciplinary action as a result. For more information read the University of Pretoria's plagiarism policy on url http://www.library.up.ac.za/plagiarism/index.htm

1.3 Uploads

The student is advised to ensure that tarballs are created properly before uploading. The *.tar.gz* created by a student should not have any sub-folders. The zip folder should **ONLY** contain the source code.

Task 1: Book Class [5]

Design a class called **Book** that holds the following information about books in a library. The class contains private variables: *`title, author, ISBN, genre;`* all of type *string*.

In the program, implement a constructor that takes in the *`title, author and ISBN,`* in this order.

Your program needs to overwrite the `ostream` **`operator`**`<<`, so that it prints the *`title, author and the ISBN`* in that order.

The following is a sample format for the modified *ostream*:

```
Starting Out with C++ - Tony Gaddis - 987646725348
```

Write appropriate accessor and mutator functions for *title , author, genre and ISBN*.

A description of your functions are shown below:

- `string getTitle()` returns the title
- `string getAuthor()` returns the author
- `string getISBN()` returns the ISBN
- `string getGenre()` returns the Genre
- **`void`** `setTitle(string a)` modifies the variable title to *a*
- **`void`** `setAuthor(string a)` modifies the variable Author to *a*
- **`void`** `setISBN(string a)` modifies the variable ISBN to *a*
- **`void`** `setGenre(string a)` modifies the variable genre to *a*

Create a makefile that compiles your program. Your program should have two files called **book.cpp** and **book.h**.

`Note: You can create your own main.cpp to test your program.`

On completion, create a tarball containing the **makefile**, **book.cpp** and **book.h**. Upload it using the active fitchfork assignment called **Practical3 - Book Class**.

Task 2: Library Class [25]

Design a class named **Library** to simulate the inventory in a library. The books should be stored in the form of pointers to Book objects, from the first task. Initially the library should be able to hold 5 books. Note that all comparisons between Book objects should be between the pointers, and not the actual Books.

This class needs to contain the following private variables:

- `string name`
- `Book** books`
- **`int`** `librarySize`
- **`int`** `numBooks`

Implement a constructor for the Library that takes in the library name and sets it, as well as a copy constructor. The copy constructor needs to make a copy of each book in the inventory of the old library and add it to the new library.

The class must implement the following publicly overloaded operators:

- `The += operator`:
  This operator adds a book to the library. The parameter is a Book pointer. If there is no space in the library output *"Library is full!"* followed by a newline, and return. An example usage of this would be:

```
Books* book = new book("Title", "Author", "ISBN")
Library library("The Library");
library += book;
```

- `The -= operator`:
  This operator removes a book from the library. If the library is empty, simply return. The parameter is a Book pointer. An example usage of this would be

```
Library -= book;
```

- `The assignment operator.`

- `The post-increment operator`:
  This operator increases the amount of books the library can hold.

```
library++;
```

- `The pre-decrement operator`:
  This operator decreases the amount of books the library can hold. If the library is full, it removes the last book.

```
--library;
```

Additionally, the class needs to have the following public methods:

- `Book* getBook(string)`:
  Returns the first book with a matching title. Returns null if no such book is found.

- `bool isFull()`:
  Returns true if the library is full, and false otherwise.

- `void print()`: Prints out the inventory of the library. If there is an empty space, print out *"[Empty Space]"*. The format is provided to you:

```
Inventory of Pretoria Public Library:
==================================
1. Inkspell - Cornelia Funke - 9780439896412
2. The Name of the Wind - Patrick Rothfuss - 9780756404741
3. Ender's Game - Orson Scott Card - 9788617354691
4. Dune - Frank Herbert - 9781587464687
5. [Empty Space]
==================================
```

  If the library is empty:

```
Inventory of Pretoria Public Library:
==================================
EMPTY
==================================
```

Create a tarball containing the **makefile**, **Library.cpp** and **Library.h**. Upload it using the active fitchfork assignment called **Practical3 - Library Class**.

1.4 Task 3: Librarian Information Class [15]

The library ensures that an updated database is always available for all library users. Hence, it's not very safe to allow anyone to add and remove books from the library without the knowledge of the librarian. It would be wise to have an interface class that allows one to interact with the Library without changing it directly.

Change the Library class from Task 2 so that the `+=, -=, =, ++, -- operators are` **`private`** `instead of` **`public`**.
Declare Librarian to be a friend in the Library class (so that librarian may access these private methods). Take note that your library.h will be overwritten if uploading to fitchfork, so make sure that Librarian only interfaces with library via the specified methods and overloaded operators in task 2.

The following private variable should be added to the Librarian class.

- `Library &lib`:
  The Library that the Librarian 'works' at. Note that member reference variables **need** to be initialized at construction.
- **`const`** `string name`:
  The librarians name.

The following public methods should be added to the Librarian class. All output from these methods should be prepended by the librarian's name (See the example output for more information):

- `Book* lendBook(String name)`:
  Lends a book from the Library, returning a pointer to the book. Remove that book from the Library. If the book is found, print out a message which includes the title of the book. If the book is not found, print out *"Sorry, we don't have that book!"*.

  Example output:

  ```
  Mary: Sorry, we don't have that book!
  John: Here is the Harry Potter, we hope you enjoy it!
  ```

- **`void`** `returnBook(Book* book)`:
  Returns a book to the Library, and adds it back to the inventory. Prints out a message including the title of the book. If the Library is full, increase it's size and then add the book.

  Example output:

  ```
  Sandy: Thanks for returning Harry Potter!
  ```

Create a tarball containing all the source code of the system and upload it using the active fitchfork assignment called **Practical3 - Librarian Class**.