Middle East Technical University Department of Computer Engineering

CENG 242

Algorithms

Spring 2019-2020

Mini Homework 2

C++

Start Time: 20 May 2020, 09:00
End Time: 21 May 2020, 09:00

1 Objectives

The objective of this mini homework is to assess and evaluate your object oriented programming skills on C++ programming language.

Keywords: *constructor, copy constructor, destructor, assignment operator, operator overload, inheritance, polymorphism, abstraction*

2 Problem

In this mini homework, you are going to implement a bookstore-book scenario. Basically, there will be 2 classes `BookStore` and `Book` and 2 more classes `HardCopy` and `SoftCopy` which are derived classes of the base class `Book`.

- `BookStore` is already implemented and totally given to you. Together with the instructors, we thought that it may get attract you, and instead of totally removing it from the mini homework we decided to give to you. The details of the properties and data structures about `BookStore` is explained in Section 3.4 for those who may wonder. However, you can skip that section if it does not attract you.
- Also, for the other 3 classes `Book`, `HardCopy` and `SoftCopy`, most of the methods are given to you as implemented in order not to make you confuse much. You are going to implement only a few methods in those 3 classes.
- This time, header files are also totally prepared for you. You are not allowed to change anything on the header files. The only thing that you need to do is implement your code in Book.cpp, HardCopy.cpp and SoftCopy.cpp files.
- This pdf gives you only the general purpose of the classes. Some visuals for you to understand better are also given in the pdf. Method implementation details are given in both here and in the header files.
- This pdf includes 6 pages where the first 3 pages are only the explanations and the last 3 pages are pictures.

3 Specifications & Implementation

3.1 Book.cpp

This is the most basic class of the assignment. A book object is defined by two things: ISBN and its price. You can consider ISBN as the identity number given for each book uniquely. Prototype of the `Book` class is given in the header file. Note that `Book` is an abstract class.

Not Important Details: It is also based on a singly linked list data structure and holds a 'next' pointer to point its next element. **However, there is nothing that you need to do with 'next'. All the parts related with 'next' are implemented.** This "linked-list-property" is added for BookStore to hold its books in an ordered way according to their ISBNs. Moreover, when a new book is added to BookStore, the ordering operation becomes much more efficient.

3.2 HardCopy.cpp

This is a derived class of the base class `Book`. It represents the usual books **sold physically** in bookstores. Prototype of the `HardCopy` class is given in the header file.

3.3 SoftCopy.cpp

This is a derived class of the base class `Book`. It represents the books **sold online**. This class holds the data in its pages.

It is based on a **binary tree structure** as follows: Each SoftCopy object contains two subtrees which are also a SoftCopy such that one of those subtrees represents the initial half of the parent SoftCopy whereas the other one represents the second half of the parent SoftCopy. "Half of a softcopy" is measured according to the number of pages it has. Each leaf of the tree represents a SoftCopy object containing a single page. See Figure 1.

- **UploadPage() Method:**
  Initially, during the construction of the softcopy, content of its pages are not given. Only its initial page id and the final page id are given other than its ISBN and price. For the pages that are asked to be uploaded into a SoftCopy object, `void UploadPage(const char* pageContent, int pageID)` method is called by supplying the page content data and the id of the page as arguments. Through this method, the corresponding leaf SoftCopy is constructed together with the branch that going down from root to leaf. **Unless a page is uploaded, construction of the branch that going down deep to the leaf softcopy is not completed.** Please examine the figure given in Figure 3. **Note:** In terms of the construction process explained just above, this class resembles to `Squadratic` class in Homework-4.

- **operator+() Method():**
  Sometimes, a subtree of a SoftCopy object can be defined independently from the main SoftCopy object. A **SoftCopy** b representing a subtree of some other **SoftCopy** R can be added into R through the summation operator overload (operator+()). See the Figure given in Figure 4. **Note:** A SoftCopy R and its subtree b will never be given as intersected. Namely, if a page with id i is previously uploaded to R, then you can be sure that b does not include a page with id i, or vice versa, namely if a page with id is previously uploaded to b, then you can be sure that R does not include a page with id i.

– **Display() Method():**
Last thing to say, it is possible to display the content of the pages uploaded via `const char* Display(int from, int to) const` method. When you supply the page id interval starting from *'from'* to *'to'* to the method, it returns content of the uploaded pages in the given interval. See the Figure given in Figure 2. Note that there will be no new line character at the initial and at the end of the output string. Only, there will be new line character between the strings The prototype of this class is given in the header file.

3.4 BookStore.cpp

This class represents a bookstore. All the books in this class are increasingly ordered with respect to their ISBN's and all books are linked to each other through a linked-list structure. It is possible to add new books to the object of this class. Also, for a BookStore object, it is possible to get the book list in a given ISBN interval. The prototype of this class is given in the header file.

4 Grading

* Book.cpp
  · `virtual void Discount() = 0;` .... 10 points
  · `bool operator<(const Book&) const;` .... 15 points
* SoftCopy.cpp
  · `Copy Constructor` .... 10 points
  · `Assignment Operator` .... 10 points
  · `SoftCopy& SoftCopy::operator+(const SoftCopy& softcopy) const;` ... 25 points
  · `void SoftCopy::UploadPage(const char* pageContent, int pageID);` .. 15 points
  · `const char* SoftCopy::Display(int from, int to) const;`.................. 15 points
* Memory leak .... 20 points penalty from total grade

5 Testing & Submission

* You can test your code interactively using CengClass running environment.
* Make sure you submit your code on CengClass.
* Given test cases are provided for your convenience, and will change later. Hence, the evaluated grade is NOT your final grade.

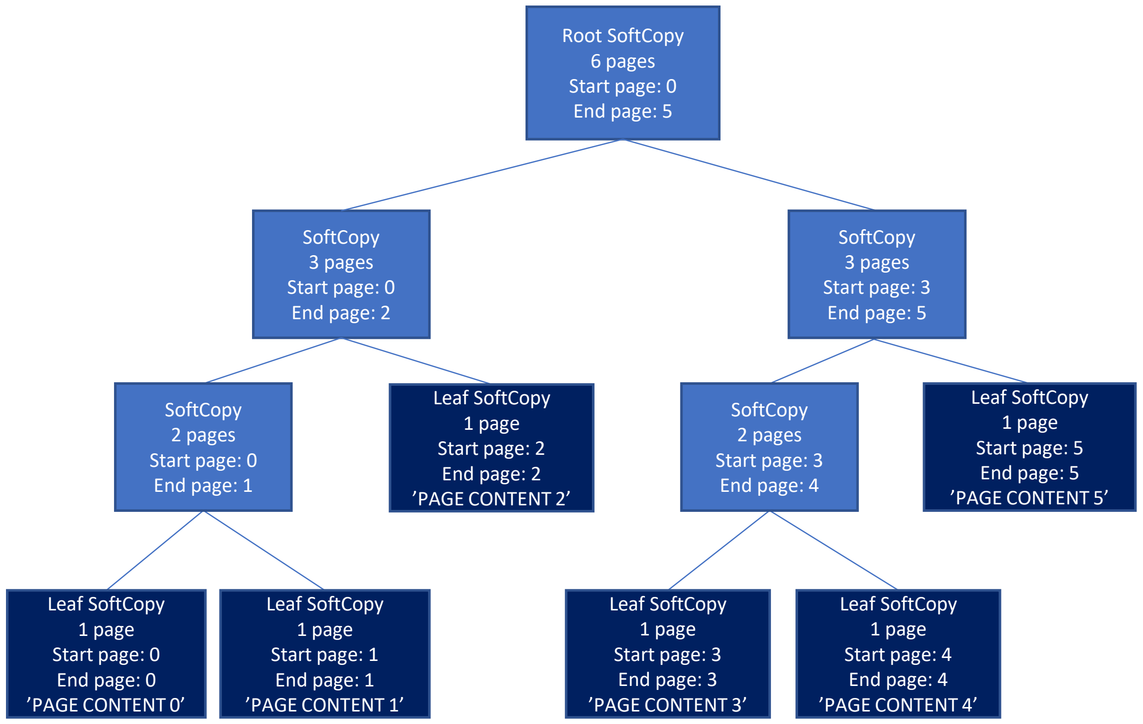


Figure 1: A SoftCopy object whose all pages are uploaded. Note that this SoftCopy includes 6 pages at total. While calculating its half, if its size is an odd number, say 2N+1, then construct its left child to have N+1 pages and its right child to have N pages as illustrated in the figure.

```
PAGE CONTENT 1
PAGE CONTENT 2
PAGE CONTENT 3
PAGE CONTENT 4
```

(a) *Display(1, 4)* call for the SoftCopy given in Figure 1

```
PAGE CONTENT 1
PAGE CONTENT 5
```

(b) *Display(0, 5)* call for the **last version** of the SoftCopy given in Figure 3

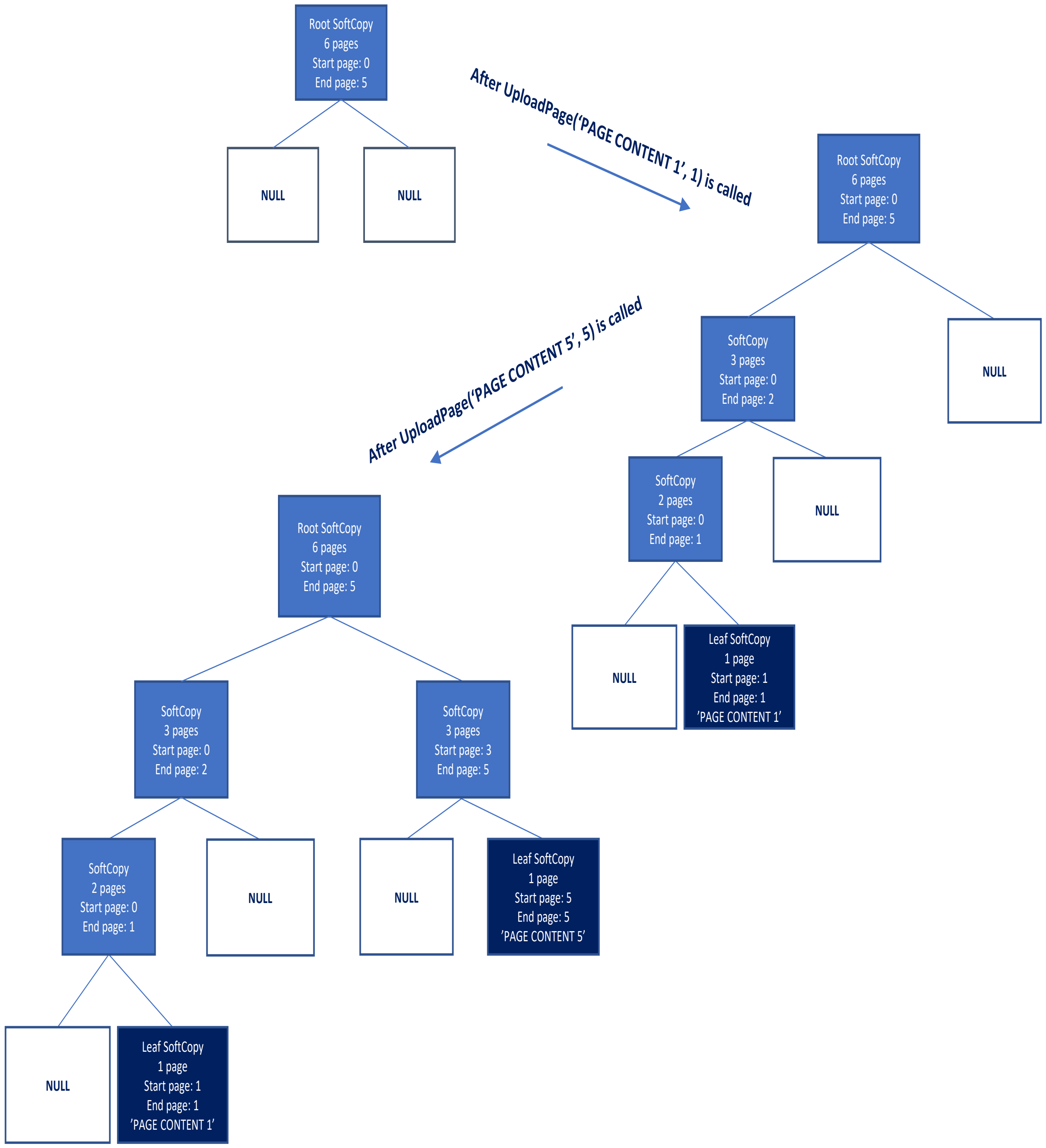
```
PAGE CONTENT 2
```

(c) *Display(2, 2)* call for the SoftCopy given in Figure 1

← There is nothing, you should return an empty string

(d) *Display(3, 4)* call for the **last version** of the SoftCopy given in Figure 3

Figure 2: Some examples from `Display()` method calls. Note that there will be no new line character at the initial and at the end of the output string. Only, there will be new line character between the strings.



Root SoftCopy 6 pages Start page: 0 End page: 5
NULL
NULL
After UploadPage('PAGE CONTENT 1', 1) is called
Root SoftCopy 6 pages Start page: 0 End page: 5
SoftCopy 3 pages Start page: 0 End page: 2
NULL
SoftCopy 2 pages Start page: 0 End page: 1
NULL
NULL
Leaf SoftCopy 1 page Start page: 1 End page: 1 'PAGE CONTENT 1'
After UploadPage('PAGE CONTENT 5', 5) is called
Root SoftCopy 6 pages Start page: 0 End page: 5
SoftCopy 3 pages Start page: 0 End page: 2
SoftCopy 3 pages Start page: 3 End page: 5
SoftCopy 2 pages Start page: 0 End page: 1
NULL
NULL
Leaf SoftCopy 1 page Start page: 5 End page: 5 'PAGE CONTENT 5'
NULL
Leaf SoftCopy 1 page Start page: 1 End page: 1 'PAGE CONTENT 1'

Figure 3: An example of successive `UploadPage()` calls

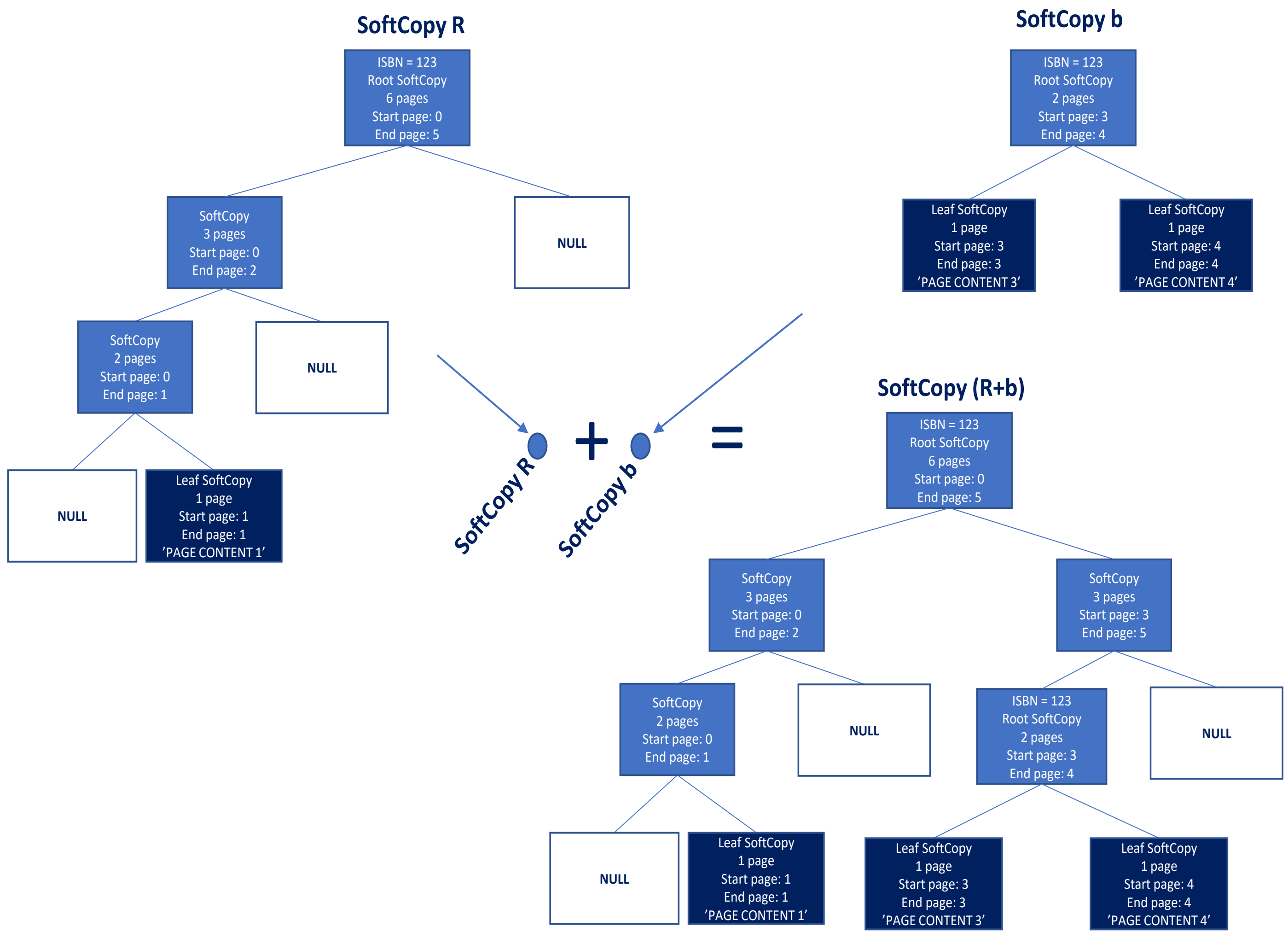


Figure 4: An example summation, `operator+()` call