[LW] Rule of Three Practice

For this labwork, the goal is to get proficiency with classes that have dynamically allocated attributes. We will provide you with a starter code that is an almost-complete program. You will declare and implement a few member functions, including the *Rule of Three* ones.

* Background material: The Rule of Three refers to the member methods destructor, copy constructor, and copy assignment operator. You can refer to Zybook sections 10.2-10.5 or to lecture material.
* Refer to [**Grading**](#_3znysh7) for details of satisfying labwork completion. Work with others in your group on the labwork, but submit individually on Mimir.

# Objectives

* Correctly code the Rule of Three member functions: destructor, copy constructor, copy assignment operator. Understanding these methods and the difference between “shallow copy” and “deep copy” is important for a successful completion of the current and future homework/exam assignments.
* Correctly code the operator<< member function of a class
* Practice (again) increasing capacity of dynamically allocated arrays.

This is not a time-consuming assignment. It has been conceived to give you time during the lab session to complete and understand the code you are writing. If you need to go back to understanding concepts such as destructors, do it now. Take the opportunity to ask questions to your labmates or the Teaching Assistant. Discuss why member functions such as the copy constructor and copy assignments are needed in one class but not in another. If you finish the lab still finding some parts of the code you wrote mysterious, ask the Teaching Assistant or your instructor during the next lecture or Q&A session.

# Grading

As stated in the syllabus, you must complete both items below to receive credit for this week's lab. **The Teaching Assistants will strictly enforce attendance that is described in this section at the start and end of your lab.**

* complete the labwork. Completion requires a minimum Mimir score of 85 points.
* attend your assigned lab

## Attendance Grade

* Your TA will mark your attendance at the start of class, and will confirm your attendance after your group shows the minimum required completed work to be checked by your TA.
* If you do not attend your lab at the start of class or if you do not receive confirmation from your TA when your group submits, then your attendance will not be recorded.

## Makeup Work

## Before you can do any make up work, you must provide your instructor with any documentation for your excused absence.

## Submission

You must download the starter code. Once you add the required methods, you can execute the application locally and submit on Mimir for unit tests. The Mimir submission requires the following files:

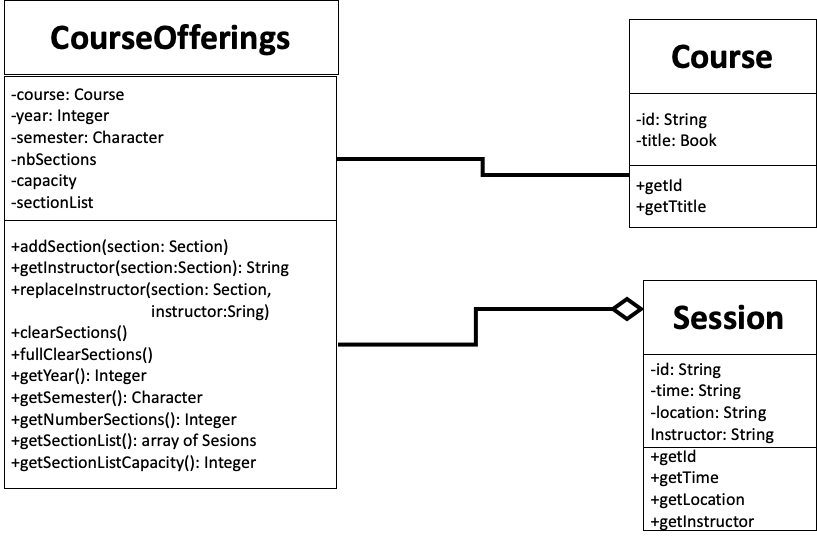
* CourseOfferings.h: The place you will add the declarations of the Rule of Three member functions. The other member functions that you need to implement are already declared in the starter code.
* CourseOfferings.cpp: The place you add your implementation of the functions that you need to implement.
* Course.h
* Section.h

You do not need to modify any of the other files in the starter code (main.cpp and the .h/.cpp files for classes Section and Course)

# Instructions

## Task 1: Get familiar with the starter code

The starter code defines the classes Course, Section, and CourseOfferings. The figure below illustrates the relationship between these classes:



Read the main() function in main.cpp and discuss with your labmates what the expected output is.

## Task 2: Implement CourseOfferings::increaseSectionListCapacity()

The file CourseOfferings.cpp specifies what the function is expected to do.

## Task 3: Declare and define the Rule of Three member functions

Notice that the class CourseOfferings has a data member that is dynamically allocated: the array sessionList is in the heap memory. Therefore, we need the Rule of Three methods:

* **The Destructor.** You will declare and define a destructor. This will avoid memory leaks.  
  Refer to Zybook Section 10.3 or lecture material for the expected signature for destructors.
* **The Copy Constructor.** You will declare and define a copy constructor. This is the code that will be executed when you initialize a new CourseOffering object from another object, as below:  
  CourseOfferings co121fall20(courseObj, 2020, ‘C’);  
  CourseOfferings co121spring21(co121fall20);

Refer to Zybook Section 10.4 or lecture material for the expected signature of the operator+ member function.

* **The Copy Assignment Operator.** You will declare and define a copy assignment operator (operator=). This is the code that will be executed when doing assignments of CourseOffering objects such as below:  
  CourseOfferings co1( /\*...\*/);  
  CourseOfferings co2(/\*...\*/);

co1 = co2; // this will automatically invoke course1.operator+(course2)

Refer to Zybook Section 10.5 or lecture material for the expected signature of the operator+ member function.

Add the declaration of the Rule of Three member functions to CourseOfferings.h and their implementation to CourserOfferings.cpp. You can submit the files to Mimir for unit tests. You should implement the destructor first, otherwise the memory leak test for the copy assignment constructor will fail even if your copy assignment method is correct.

## Task 4: Implement the missing member functions (declared but not defined)

The starter code declares the following methods without providing an implementation:

* CourseOfferings::clearSections()

Public member function that updates sectionList to have zero size.

* CourseOfferings::fullClearSections()

Public member function that fully deallocates sectionList from heap memory.

The expected functionality is described in the cpp file CourseOfferings.cpp. You can submit the files to Mimir for unit tests.

## Task 3: Implement the operator<< function

The starter code came with the declaration of the operator<< for doing output of CourseOfferings objects. You must implement it in CourseOfferings.cpp such as the output follows the format below. Text in magenta represents data from the specific object being output.

Year: 2020 Semester: C Course id: CSCE121

Course title: Introduction to Program Design and Concepts

Sections:

597 N/A web-based Michael Moore

521 TR 1:30 2:45 pm ONLINE Philip Ritchey

512 MW 5:35 - 6:50 pm ZACH 310 Paul Taele

508 TR 8:00 - 9:15 am ONLINE Dilma Da Silva