# Description

The purpose of this lab is to check your understanding of the content in lecture 5a, *Classes*.

We will focus on the material from chapter 9 of the book. The objectives are:

* How to use a preprocessor wrapper to prevent multiple definition errors caused by including more than one copy of a header file in a source-code file.
* To understand class scope and accessing class members via the name of an object, a reference to an object or a pointer to an object.
* To define constructors with default arguments.
* How destructors are used to perform “termination housekeeping” on an object before it is destroyed.
* When constructors and destructors are called and the order in which they are called.
* The logic errors that may occur when a public member function of a class returns a reference to private data.
* To assign the data members of one object to those of another object by default member-wise assignment.

# Part 1: Questions

Complete the assignment in the module *Topic 5 Lab 5a: Classes* named, "Lab 5a Questions", in Canvas.

# Part 2: Activity

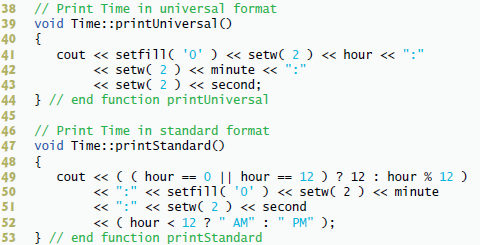
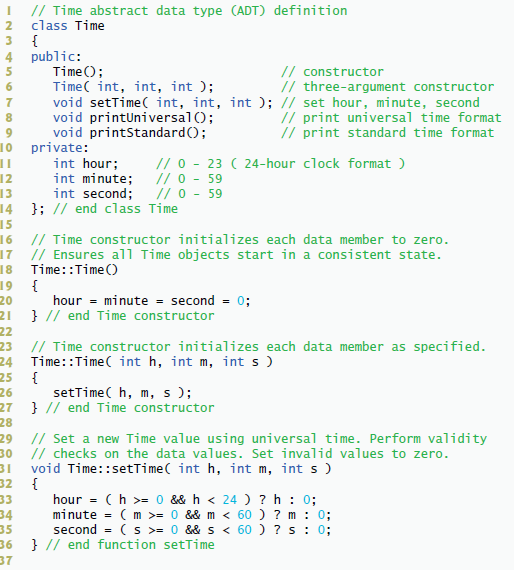
This part of the lab should be submitted using the Lab 5a Activity assignment in the *Topic 5 Lab 5a: Classes* module.

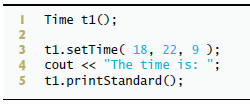
**You should complete the non-programming parts of this lab using this document.** **It is possible that some of the images and answer-boxes below might move or need to be resized while using them. Do your best to make your final document neat and organized.**

## Problem 1: Program Output

For each of the given program segments, read the code and write the output in the space provided below each program. [Note: Do not execute these programs on a computer.]

For Programming Output Exercises 1 - 3, use this class definition.



1. What is output by the following code segment? Use the definition of the class Time.

**Answer**

**The time is: 6:22:09 PM**

1. What is the output of the following program segment?

problem 1-2

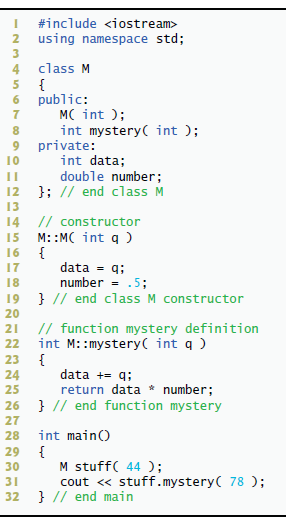
**Answer**

**3:04:05 AM**

**03:04:05**

**00:03:05**

1. What is output by the following program? Use the Time class.



**Answer**

**61**

1. What is output by the following program?

problem 1-4

**Answer**

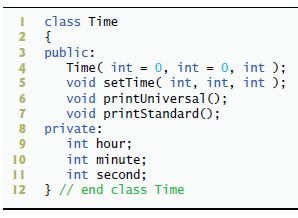
**22**

**32**

## Problem 2: Correct the Code

For each of the given program segments, determine if there is an error in the code. If there is an error, specify whether it is a logic error or a compilation error, circle the error in the program, and write the corrected code in the space provided after each problem. If the code does not contain an error, write “no error.” [Note: It is possible that a program segment may contain multiple errors.]

1. The following should define class Time:



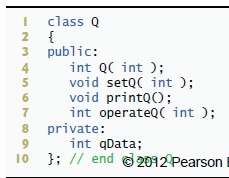


**Answer**

**Compilation error**

**};**

1. The following code defines class Q:



**Answer**

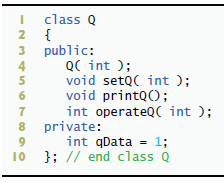
**Logic error**

**public:**

**Q( int );**



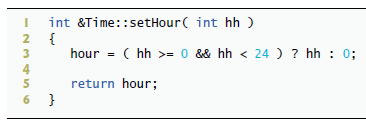
1. The following is another version of class Q’s definition:



**Answer**

**No error**

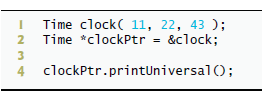
1. The following defines setHour, a member function of the Time class.



**Answer**

**No error**

1. The following code should call member function printUniversal of the Time class





**Answer**

**Compilation error**

**\*clockPtr.printUniveral();**

### Problem 3: Programming

The Bucket data structure:

Collections (or containers) are a very important part of programming. A "bucket" data structure is a container that follows the form and function of many data structures. Its purpose is to collect and store any type of values, your bucket will store only doubles. Using simple operations, like add, remove, count, etc., one can simply store and retrieve large quantities of values using a small amount of code.

The bucket is an unordered collection. Simply put, an unordered collection of elements are stored in a way that their order cannot be predicted. Other data structures like stacks and queues store their elements in a predictable order (these are known as ordered collections), buckets do not.

Implement the *bucket* class from the header file provided. The header will be called, "bucket.h". It contains a variable of type size\_t to show the default size of the bucket. Initially, the array is declared at this size. Because Bucket is a class it must be defined in bucket.h and bucket.cpp.

### Requirements

* The bucket is a utility class. It is used by client code to simply store and retrieve items of double type. **There is no console input or output in the bucket class.**
* The bucket dynamically resizes when it gets full. Its array is created on the heap. The array should double in size when an *add* operation is called but the bucket is full. Be sure to deallocate the old array after copying its data to the new array. Use *new* to create the new array and *delete[]* to deallocate the old one.
* You must implement the bucket in two files: the header file (.h) will contain the declaration of the bucket (tis file is provided), and the definition file (.cpp) will contain the definition of the bucket’s functions.
* **Do not add to or change the public members of the class.** You can add private utility functions and data members but do not alter the public section of the class.
* Do not include your own header files in this class. This class should be a stand-alone class that requires only C++ standard library includes.
* Read the pre and post conditions for each member function in the public section of bucket.h to understand its responsibilities.
* You will write a test program called, "test\_bucket.cpp" to thoroughly test your Bucket class' functions.

# What to Submit for Lab 5 Activity

* This document completed.
* bucket.h
* bucket.cpp
* test\_bucket.cpp