Object Oriented Programming & Design
CS 244 - 001
Department of Physics and Computer Science
Medgar Evers College
Exam 1

Instructions:

- **The exam requires completing a set of tasks within 80 minutes.**
- **Case: You are tasked with designing a digital alarm clock application that includes an interchangeable 12-hour and 24-hour display formats, storage for the current time (hour & minute), alarm time (hour & minute), display format, and alarm volume.**
  Additionally, it can set time and alarm (hour and minute separately), toggle between display formats, turn the alarm on and off, adjust the volume (range [0,100]), use a scrollbar to increment/decrement hour, minute, and volume sequentially. It ensures hour and minute scrolls are modular (looping at boundaries); whereas, the volume scoll is non-modular.
- **Your objective is to create portions of a header file named '`Clock.h`' that defines a class *Clock* representing the above application. and answer object-oriented programming conceptual questions.**
- **Questions are cumulative.**
- **Notes are not allowed.**
- **Cheating of any kind is prohibited and will not be tolerated.**
- **Violating and failing to follow rules will result in an automatic zero (0) for the exam.**

TO ACKNOWLEDGE THAT YOU HAVE READ AND UNDERSTOOD THE INSTRUCTIONS ABOVE, PRINT YOUR NAME AND THE DATE ON BOTH THIS SHEET AND THE BLUE BOOK

Name: _____ **Date:** _____

Grading

Problem	Maximum Points	Points Earned
01	1	
02	1	
03	1	
04	1	
05	1	
06	2	
07	1	
08	1	
09	2	
10	1	
11	2	
12	1	
13	1	
14	1	
15	2	
16	1	
Total	20	

1. **Write the header guard for '`Clock.h`' using `CLOCK_H` and include the libraries *iostream*, *string*, *iomanip*, and *sstream*.**

2. **Define the concept of abstraction in object-oriented programming.**

3. **Explain the difference between static and instance fields.**

4. **Declare**
   - **private integer array field named *times* with size 4 that represents the time hour, time minute, alarm hour, and alarm minute, respectively.**
   - **private integer field named *volume*.**
   - **private Boolean array field named *switches* with size 3 that represents format (true for 12 hour, false for 24 hour), alarm state (true for on, false for off), and mode (true for alarm view, false for time view), respectively.**
   - **private static constant string array field named *meridiems* initialized to {`"AM"`, `"PM"`}.**
   - **private static integer field named *count* initialized to zero.**

5. **Explain the purpose of a getter method and provide its general syntax.**

6. **Define the getter methods named `hour()`, `minute()`, and `volume()`.**
   - **`hour()` method returns the hour of the mode**
   - **`minute()` method returns the minute of the mode**
   - **`volume()` method returns the *volume*.**

7. **Define the default constructor of *Clock* that initializes *times*, *volume*, and *control* to {0, 0, 6, 0}, 30, and {false, false, false}, respectively, and increments *count* by 1.**

8. **What is a namespace? Provide its declaration syntax.**

9. **Define the assignment operator for *Clock*.**

10. **Define a static method named `clocks()` that takes no parameters and returns *count*.**

11. **Define the string constant method `toString()` that takes no parameters and returns a string in the format**

$$\begin{cases} \texttt{"(}v\texttt{)}\ h\texttt{:}m\ z\texttt{"} & \text{if format is 12 hour} \\ \texttt{"(}v\texttt{)}\ h\texttt{:}m\texttt{"} & \text{if format is 24 hour} \end{cases}$$

    where v is `'T'` if the mode is time view or `'A'` if the mode is alarm view, h is the hour of the mode as two digits, m is the minute of the mode as two digits, and z is the times corresponding meridiem.

12. **Define the destructor for *Clock* that decrements *count* by 1. Explain the purpose of a destructor.**

13. **Define a void method named `alarm()` that takes no parameters and toggle the alarm state.**

14. **What is a constant method? Explain its constraints.**

15. **Define the void setter methods `adjustTime()` and `adjustVolume()`.**
   - **`adjustTime()` takes two Boolean parameters and adjusts the specified component of the mode in the indicated direction given that the component (first parameter) implies hour if true and minute if false, and the direction (second parameter) implies increment one unit if true and decrement one unit if false.**
   - **`adjustVolume()` takes a Boolean parameter and adjusts *volume* in the indicated direction if possible given that the direction (the parameter) implies increment one unit if true and decrement one unit if false.**

16. **What is the this pointer, and in what situations is it required? Please provide at least one example.**