Lab 10

Comp 295 Introduction to C/C++

Task 1

Objective: The goal of this lab exercise is to design and implement a multi-file program using the C++ object-oriented paradigm, particularly inheritance.

You will create a simple simulated environment where there are different types of media files - audio, video, and text. You are tasked to create appropriate classes for these media types, where the common features are included in a base class named `Media`, and the unique features are implemented in the derived classes `Audio`, `Video`, and `Text`.

Task:

1. Implement a base class `Media` that stores the following information and operations common to all media types:
   - A title (string).
   - A creator or author (string).
   - A method to display the basic information of the media file.
2. Implement a derived class `Audio` that includes the following additional information and operations:
   - Audio quality (in Kbps).
   - A method to display the complete information of the audio file, including the information from the base class.
3. Implement a derived class `Video` that includes the following additional information and operations:
   - Video resolution (in pixels).
   - A method to display the complete information of the video file, including the information from the base class.
4. Implement a derived class `Text` that includes the following additional information and operations:
   - Number of words.
   - A method to display the complete information of the text file, including the information from the base class.
5. Your program should be organized using a three-file structure:
   - A header file (`media.h`) that contains the class definitions.
   - A source file (`media.cpp`) that contains the class implementations.
   - A main source file (`main.cpp`) that contains the `main()` function and demonstrates the functionality of your classes.

Requirements:

1. Your classes should be implemented using appropriate C++ syntax and style. This includes proper use of inheritance, encapsulation (private and public members), and function overriding (if necessary).

2. Your `main()` function should create instances of each media type, populate them with appropriate data, and display that data to demonstrate that your classes are working correctly.
3. Comment your code adequately to explain your logic and any decisions you made in your implementation.

Deliverable:

You need to submit three C++ files (`media.h`, `media.cpp`, and `main.cpp`). Your solution will be evaluated based on its correctness, code organization, use of object-oriented principles, and the appropriateness of the test data chosen to demonstrate the functionality.

Task 2

Objective: The aim of this lab exercise is to design and implement a multi-file program using the C++ object-oriented paradigm, specifically inheritance, polymorphism and dynamic memory management with dynamic arrays. You will still create a simulated environment with different types of media files - audio, video, and text. Moreover, you are now required to create an additional `MediaLibrary` class to manage these media files.

Task:

1. Implement the `Media`, `Audio`, `Video`, and `Text` classes as described in the previous task.
2. Implement a class `MediaLibrary` which stores a dynamic collection (using a dynamically allocated array) of pointers to `Media` objects. The `MediaLibrary` should provide the following methods:
   - `addMedia`: a method that takes a pointer to a `Media` object and adds it to the library. The library should be able to store a pre-defined capacity of media objects.
   - `displayLibrary`: a method that displays the information for all `Media` objects in the library using dynamic binding.
3. Your program should be organized using a three-file structure:
   - A header file (`media.h`) that contains the class definitions.
   - A source file (`media.cpp`) that contains the class implementations.
   - A main source file (`main.cpp`) that contains the `main()` function and demonstrates the functionality of your classes.

Requirements:

1. You should use inheritance and dynamic binding to ensure that the correct `display` method is called for each media type when using the `MediaLibrary::displayLibrary` method.
2. You should demonstrate correct memory management. Any memory allocated with `new` must be deallocated with `delete` when it is no longer needed. This is especially important for the `MediaLibrary` class, which should deallocate the `Media` objects when they are removed from the library or when the `MediaLibrary` is destroyed.
3. As before, your code should be implemented using appropriate C++ syntax and style, and should be adequately commented.

Deliverable:

You need to submit three C++ files (`media.h`, `media.cpp`, and `main.cpp`). Your solution will be evaluated based on its correctness, code organization, use of object-oriented principles, memory management, and the appropriateness of the test data chosen to demonstrate the functionality. The additional complexity of managing dynamic arrays will also be considered in the evaluation.

This is how your `main.cpp` file will look like:

```
#include <iostream>
#include "media.h"

int main() {
    MediaLibrary myLibrary(3);
    myLibrary.addMedia(new Audio("My Audio", "Audio Creator", 320));
    myLibrary.addMedia(new Video("My Video", "Video Creator", "1920x1080"));
    myLibrary.addMedia(new Text("My Text", "Text Creator", 1000));

    std::cout << "Displaying Media Library:" << std::endl;
    myLibrary.displayLibrary();

    return 0;
}
```