Lab 3: Movie Archives

Purpose of this Lab:

For this lab you will work on writing classes, using accessor functions, dynamic arrays, constructors, and accessing/modifying the state of objects in your program.

Introduction to the Lab :

You will write a program that can represent the properties of a typical blu-ray movie as an object so that you can finally organize your movie collection or at least pretend to do so.

Program instructions:

Here is the starter code **Lab3StarterCode.zip (https://canvas.eee.uci.edu/courses/46984/files/19775695?wrap=1)** (https://canvas.eee.uci.edu/courses/46984/files/19775695/download?download_frd=1)

Complete the program by implementing the functionality defined here and in the starter code, and upload your completed .ccp and .h files to Gradescope.

- A Movie class consisting of the following:
  - private:
    - int - duration of the movie (in minutes)
    - int - year the movie was released
    - int - number of actors in the movie
    - double - price of the movie (in USD)
    - string - title of the movie
  - public:
    - Actor* actors
      - An array of objects representing the actors starring in the movie
    - Movie()
      - Constructs a Movie object with default parameters
    - Movie(duration of the movie, release year, number of actors, movie price, movie title)
      - Constructs a Movie objects with all the parameters passed in
    - Movie(movie)
      - Constructors a Movie object by *deep* copying all the attributes of the movie object passed in.
    - ~Movie()
      - Destroys the Movie object, making sure to not creating any memory leaks
    - void setMovieTime(duration of the movie)
    - int getMovieTime()

        - void setMovieYearOut(release year)
        - int getMovieYearOut()
        - void setNumberOfActors(number of actors)
        - int getNumberOfActors()
        - void addActor(first name, last name, year of birth)
            - Adds an actor to the actors array by constructing an Actor object with the attributes passed in
        - void setMoviePrice(movie price)
        - double getMoviePrice()
        - void setMovieTitle(movie title)
        - string getMovieTitle()
        - void printMovieInfo()
- An Actor class consisting of the following:
    - private:
        - string - first name of actor
        - string - last name of actor
        - int - birth year of actor
    - public:
        - Actor()
            - Constructs an Actor object with default parameters
        - Actor(first name, last name, year of birth)
            - Constructs an Actor object with the parameters passed in
        - void setFirstName(first name)
        - string getFirstName()
        - void setLastName(last name)
        - string getLastName()
        - void setBirthYear(year of birth)
        - int getBirthYear()
- A Lab3 cpp consisting of:
    - The main function that starts your program.
        - This program should only create objects and call functions from the other classes. Do not create functions inside this file
        - testing() function to the main program, which is called to test functionality of Movie and Actor classes instead of main. You can simply copy-paste your tests in this function or define multiple test functions like this one to do unit testing on every functionality separately.

          This to try to make two things easier to catch: segmentation faults and bugs with the destructor. Modern compilers employ many tricks (e.g auto zero initialization (https://docs.microsoft.com/en-us/cpp/cpp/initializers?view=msvc-160#zero-initialization) , intelligent memory freeing, etc) to try to do their best to not let your code crash. However, these tricks do not always work and can create bugs that don't affect your code in your main(), but crash the autograder, which tests your code outside of your main() function. While having a separate test function doesn't completely address this problem, we think that it are still much better than relying exclusively on main() for testing. The test function is purely used for your to test your program.

Important details:

- Please make sure your public members (functions and variables) in both classes have the same signature as they have above. The autograder will call them them when it tests your code.
- The autograder will not call your main function defined in your Lab 3 file, and thus you are not required to prompt user for keyboard input. Instead, use your main function to test the correctness of your code as necessary.
- To test your code, you may try doing the following:
  - Constructing the same Movie using all the constructors
  - Constructing an Actor object using all the constructors
  - Setting fields using the appropriate accessor functions
  - Printing out the Movie's information (including the actors' information)
  - Testing your copy constructor by doing the following:
    - Creating another movie that is a copy of your original Movie object
    - Printing out the fields of the copied object to the console, changing the attributes of one of the objects, and printing both of them out again
- Below is the expected format of the printMovieInfo() function (there is a space after each punctuation, *except* "Actors:", the capitalization matters, and every line is terminated with an endl):

```
Title: {movie title}, Published in: {year}
Duration of Movie: {movie duration} minutes
Price: ${price}
Actors:
{actor 1's first name} {actor 1's last name}, {actor 1's birth year}
...
{actor n's first name} {actor n's last name}, {actor n's birth year}
```

  - Here is an example of the movie The Dark Knight:

```
Title: The Dark Knight, Published in: 2008
Duration of Movie: 152 minutes
Price: $14.99
Actors:
Christian Bale, 1974
Heath Ledger, 1979
```

Aaron Eckhart, 1968
Michael Caine, 1933
Maggie Gyllenhaal, 1977