School of Computing Sciences

**Module:** CMP-5015Y Programming 2
**Assignment:** Summative Coursework 2 - Offline Movie Database

**Set by:** Dr Gavin Cawley (g.cawley@uea.ac.uk)
**Date set:** 24th February 2020
**Value:** 30%

**Date due:** 29th April 2020 3pm
**Returned by:** 27th May 2020
**Submission:** evision

### Learning outcomes

The aim of this assignment is for the student to gain experience in the design and implementation of a relatively simple application in the C++ programming language. On successful completion of this exercise, the student will have reinforced a basic understanding of the concepts of classes and objects and will be familiar with the basic syntax of C++ programming constructs used to implement classes (including operator overloading) and have experience with stream-based I/O. The student will also have opportunities to explore more advanced elements of the C++ programming language, such as templates and lambdas.

## Specification

### Overview

You are required to implement and test a C++ program to load the details of a collection of films (or "movies", both terms may be used interchangeably) from a file, `films.txt` (available from BlackBoard), and provide answers to a small set of simple database queries. All questions regarding the specifications **must** be asked via the appropriate discussion board on the module volume on BlackBoard.

### Description

The file `films.txt`, available on BlackBoard, contains information describing a number of movies, including the title, the year in which it was released, the certificate, (e.g. "R" or "PG-13", indicative of the audience for which the film is considered suitable), the *genres* describing the film, its duration (in minutes) and the average ratings supplied by users. The program must load the information in this file into an appropriate collection of objects, so that database queries could be answered without re-reading the file. The program must consist of at least the following three modules:

- `Movie.h` and `Movie.cpp` - A `Movie` object describes the information stored about a particular film, such that there will be a separate `Movie` object for each film held in the database. The class should have a suitable collection of constructors, accessor methods etc. and the stream I/O and relational operators should be implemented.

- `MovieDatabase.h` and `MovieDatabase.cpp` - A collection of `Movie` objects, one for each film described in the data file. The class should provide overloaded I/O operators for reading the data from file and displaying the database on the terminal and for answering the database queries. Rather than writing a program that only implements the current specifications, we should write `maintainable` programs that are easily extended to answer a variety of database queries, without writing a lot of extra code (i.e. methods that answer generic queries are better than methods that answer very specific queries).

- `main.cpp` - This module contains the `main` function and uses the services provided by `Movie` and `MovieDatabase` to answer the required database queries. This should provide an easily understood top-level view of what the program actually does, without going into unnecessary detail of *how* it is done (which is *abstracted* away by the use of classes, as explained in the lectures on CMP-4008Y and CMP-5015Y).

  The `main` function must perform the following tasks:

  1. Read in the database from the file `films.txt`, using the relative path "films.txt", provided via BlackBoard (when using CLion, the program will expect to find the file in the `cmake-build-debug` directory). This is necessary to ensure that the program runs correctly using PASS.
  2. Display the entire collection of movies, arranged in chronological order. The movies must be displayed in the same format in which they appear in `films.txt`.
  3. Display the third longest *Film-Noir*.
  4. Display the eighth most recent *UNRATED* film.
  5. Display the film with the longest title.

Deciding the appropriate set of methods and functions for each module is an important element of the assignment, and sufficient thought to these design issues should be made *before* commencing implementation.

## Hints

- Tackle the problem one module at a time (I would start with `Movie.h` and `Movie.cpp` and have it more or less finished and tested before moving on to `MovieDatabase.h` and `MovieDatabase.cpp` and then construct `main.cpp` from the components already constructed). Note this is an example of "top-down design, bottom-up implementation", as discussed in the lectures on CMP-4008Y.

- For each module, it is a good idea to work out which interface methods and functions the module should implement in order to provide the services required by `main.cpp` *before* starting to code. If implementing the module proves unduly difficult, that is often because the *design* is incorrect.

- Classes should be used to provide *abstraction* and *encapsulation* (i.e. data should be made available on a "need to know" basis).

- Make sure you have a good idea of what you have to do *before* you start to write the code, and separate the task of working out what you want the computer to do from the task of how to tell the computer to do it.

- It is important to test each module of the program in isolation. Don't write all three modules and expect them all to work together first time. Bugs are often more easily detected and removed at module level, rather than systems level. Each module should have a test-harness.

- Implementation of `MovieDatabase` requires the use of a `vector` (or similar data structure), however `Movie` can be implemented and tested using material already covered on this module.

## Relationship to formative assessment

This assignment is essentially a re-implementation of the off-line movie database in Java coursework from CMP-4008Y and in C as a formative coursework on this module. This gives an opportunity to explore the differences and similarities between Java, C and C++ on the same task. This will reveal some of the advantages and disadvantages of each language and the progression of ideas in the development of this family of programming languages.

## Deliverables

You must submit a `.zip` file to BlackBoard using the submission point located in the `Summative Assessment` section. The `.zip` must contain the source files of your program (`Movie.h`, `Movie.cpp`, `MovieDatabase.h`, `MovieDatabase.cpp` and `main.cpp`). The `.zip` file must also contain a `.pdf` file consisting of the source code listings for program, in the order given above. Students are encouraged to use the template set up at `Overleaf.com` (see the accompanying guidance note available on BlackBoard). If you chose to generate the `.pdf` file in some other way, make sure that there is a title page on the front giving your registration number and BlackBoard ID, and that the code is formatted so that it is legible when printed on A4 paper and that lines are limited to no more than 80 columns.

## Resources

- `https://stackoverflow.com/` - An excellent site for finding information about specific issues relating to various programming languages, including C++. It is important however not to become too reliant on sites such as `StackOverflow`, which are great for details, but don't give the "big picture", so it is difficult to get a good understanding of programming in this way. Note that if you re-use or modify code found on-line, then you must provide a comment giving the URL and a brief explanation of what modifications have been made. Re-using code found on-line without proper attribution would constitute plagiarism.

- `https://en.cppreference.com/w/` - a website providing reference material on C++ and its libraries.

- `http://www.cplusplus.com/reference/` - another website providing reference material on C++ and its libraries.

If you become stuck, please do feel free to discuss problems with the teaching assistants during scheduled laboratory hours, or contact me via email (`g.cawley@uea.ac.uk`) for assistance outside lab hours. However, questions relating to the specifications will *only* be answered if asked via the discussion forum on BlackBoard.

## Plagiarism and Collusion

StackOverflow and other similar forums (such as the module discussion board) are valuable resources for learning and completing formative work. However, for assessed work such as this assignment, you may not post questions relating to coursework solutions in such forums. Using code from other sources/written by others without acknowledgement is plagiarism, which is not allowed (General Regulation 18).

## Marking Scheme

Marks will be awarded according to the proportion of the specifications successfully implemented, programming style (indentation, good choice of identifiers, commenting etc.), and appropriate use of programming constructs. It is not sufficient that the program merely generates the correct output. Professional programmers are required to produce maintainable code that is easy for *others* to understand, easy to debug when (rather than "if") bug reports are received and easy to extend. Relatively few marks are therefore assigned to correct operation in the marking scheme. The code needs to be modular, where each module has a well-defined interface to the rest of the program. The function of the modules should be made as generic as possible, so that it not only solves the problem specified today, but can easily me modified or extended to cater for future developments in the specification without undue cost. The code should be reasonably efficient. Marks may also be awarded for correct use of more advanced programming constructs or techniques not covered in the lectures. Students are expected to investigate the facilities provided by the C++ standard libraries.

- `Movie.h` and `Movie.cpp` - demonstrating basic grasp of implementing simple classes in C++ and stream-based I/O using operator overloading. Potential for use of enumerations. **[40 marks]**
- `MovieDatabase.h` and `MovieDatabase.cpp` - demonstrating use of generic containers/algorithms from the STL, potential for use of lambda's etc. **[30 marks]**
- `main.cpp`
  - Quality of implementation. **[10 marks]**
  - Task #1 - Sort the movies in ascending order of release date and display on console. **[5 marks]**
  - Task #2 - Display the third longest Film-Noir. **[5 marks]**
  - Task #3 - Display the eighth most recent `UNRATED` movie. **[5 marks]**
  - Task #4 - Display the movie with the longest title. **[5 marks]**

Note that this assignment is intended to assess C++ programming skills; while most C programs are also valid C++ programs, solutions using a C programming style will attract low marks (as they would not demonstrate skills not already covered by the formative course work).