COMP322: Assignment 3- Winter 2012

Due at 11:30pm EST, 22nd March 2012

1 Introduction

For this assignment, we will implement a simple movie recommender using classes in C++. We will also get some more practice using the algorithms in the standard library as well as with iterators and container classes. On the last assignment, we'll use inheritance to see how we can use multiple kinds of recommenders at the same time.

2 Getting Started

On the course website, you should download the file `SupportCodeA3.zip`. Inside here are the header files for the classes you will implement. In addition to the header files, there is a file `RatingInformation.cc` which contains the same definition using a struct as in assignment two. Finally there are a couple of very small txt sample files you can use as examples of the files you'll be reading from during the assignment.

You will be writing the .cc files that should accompany these .h files. You will also write a program with a main function inside of it.

You are free and encouraged to add private methods/functions to the .h files but you must not delete or change the prototypes of the existing ones.

For full credit on this assignment, you must not include any source code files (other than RatingInformation.cc) inside any other source files. You may include header files. This means that to compile the assignment you should write (all on one line)

```
g++ MovieInformation.cc RatingInformation.cc
          SimpleRecommender.cc RecommendationProgram.cc
```

OR

```
g++ *.cc
```

3 MovieInformation.cc

In this file, you should implement the interface `MovieInformation.h` The idea of the class `MovieInformation` is to encapsulate all information related to a movie. (We could also store things like actor names, category, etc if we wanted.)

The header declare 2 methods and one constructor. The constructor should take as input the two properties, name and id, to set. The 2 methods should simply get the values. They are declared in the header to be const because they don't change the class at all.

4 SimpleRecommender.cc

In this file, you should implement the interface `SimpleRecommender.h` The idea of `SimpleRecommender` is we use a VERY basic algorithm of assuming that all users are the same and just recommending the most popular movies. In assignment 4, we will use inheritance to see how easy it is to change our approach so that we can use a more complex algorithm and compare the results.

There are 4 required methods:

- generatePrediction : This method takes as input an int representing the userid and a string representing a movie name. It then takes the average of all ratings of that particular movie name. (You'll have to get the movieid of that particular movie name to do this.) It should return this average.

- makeRecommendations : This method takes as input an int representing userid and an int n representing the number of movies to recommend. It then, using the method `generatePrediction` chooses the n best movies for the user with id userid. None of these n movies should have already been rated by the user. It should return a `vector<string>` of the movie recommendations which is sorted according to the preference of the user. That is, the first element of the `vector<string>` should be the name of the top recommendation, the second element should be the name of the second recommendation, etc. If there are not n such elements (for example, if the user rated every movie), then your `vector<string>` should be "trimmed" appropriately. (i.e. Everything should work normally, except the output is just a smaller vector).

- loadMovieListFromFile : This method takes as input a String filename and initializes the property `movies` based on this. Each line of the file is one movie name and you can assign an id based on the line number. See `Sample.txt` for an example of what this file might look like.

- loadRatingsFromFile : This method also takes as input a String filename and initializes the property `ratings` based on the file. This file is the same as in assignment two. (You may adapt the solution from assignment two when it is posted.) See `TestFile.txt` for an example of this.

A lot of these tasks will be much simpler if you use the `algorithm` library. It will typically take a bit longer to compile things, but you'll be less likely to have bugs. As such, we will require that for at least one of these methods, you use a method in the `algorithm` library and define a class that *overloads* the `operator` method. `http://www.cplusplus.com/reference/algorithm/sort/` has one example of this with sorting.

The idea here is that many of the functions in the algorithm class require a function pointer as input. (This sounds scary but just means writing a function name without () as one of the parameters.) This function is used to determine how exactly to perform that algorithm. For example, the function `sort` expects as input a pointer to a function defining what *less than* means. The less than is then used to sort things. The function `remove_copy_if` will copy certain values from one container to another based on a boolean function.

In some cases, one can pass a pointer to a function directly without complication. However, in other cases, the function may require addition input to it. In this case, what one normally does is define a class that has as one of its properties this input. For example, the following class could be used, generally, to determine whether something matched ANY int. Without using this class, we'd have to write one function for each int which would defeat the purpose of writing functions.

```
class CheckMatching
{
    private :
        int n;
    public :
        CheckMatching(int number) {
            n = number;
        }

        bool operator()(const int& number )
        {
            return n == number;
        }
};
```

Now we can use this class with an stl algorithm as follows:

```
vector<int> numbers;
...
vector<int> results;
...
CheckMatching three(3); //used to check if something matches with 3
remove_copy_if(numbers.begin(), numbers.end(), results.begin(), three);
```

5 RecommendationProgram.cc

Here you are required to write a main function. Your main function should do the following:

1. Ask the user to enter a filename to load ratings from
2. Ask the user to enter a filename to load the movie list from
3. Initialize a SimpleRecommender object based on these files.
4. Your program should then have a while loop that continues until the user types -1. The program should read for the next number the user types and generate predictions for that user. It should print a list of the movies for that user. If no such movies exist either because the user doesn't exist or because the user rated all movies, your program should print `No recommendations found!`

6 Submitting your assignment

What To Submit

`MovieInformation.h`
`MovieInformation.cc`
`RatingInformation.cc`
`SimpleRecommender.h`
`SimpleRecommender.cc`
`RecommendationProgram.cc`
`Confession.txt` (optional) In this file, you can tell the TA about any issues you ran into doing this assignment. If you point out an error that you know occurs in your problem, it may lead the TA to give you more partial credit. On the other hand, it also may lead the TA to notice something that otherwise he or she would not.