**MovieExplore Popularity Algorithm**
**Software Engineering 301**
**Instructor: J. Sillito**
**Developers: Gregg Lewis + Kyle Milz**

**The Problem: Determining How To Define Popularity**
The first problem to tackle was defining the term popularity. One might say a movie is popular because a lot of people have rated it (or, more intuitively, one might say that a movie is not very popular because very few people have rated it). One might also say that popularity is a function of the ratings assigned to the movie. In truth, the answer lies somewhere in between.

**The Approach:**
The approach that we chose was to assign a point value to each rating (currently the rating itself, eg. a rating of 3 is worth 3 points, a rating of 5 is worth 5 points), and find the sum of these point values. Thus a movie with 10 ratings, with each rating being a 5/5 would have a total score of 25. This is done for all movies, and the movies are then sorted according to this point score, with preference being given to movies with fewer ratings in the case of ties. The reasoning behind this is that a movie which 50 people rated 5/5 (for a total score of 250) is probably more popular than a movie which 250 people rated 1/1 (for the same total score). In the case of a tie in both score and number of ratings, preference is given to the movie whose title comes first alphabetically. Once the movies are in order of popularity, all that remains is to convert their popularity into an integer score out of 100. This is currently done simply by dividing a movie's total point value by the maximum point value of any movie, multiplying the result by 100, and finally rounding to the nearest integer.

**Advantages to This Approach:**
The main thing that this approach has going for it is its simplicity. It adequately captures for both the number of ratings a movie has as well as the value of each of those ratings. Its current implementation is also fairly quick, as all of the ratings are simply totaled and stored as an integer, eliminating the need to do any further iteration over them. (This, of course, may change in the future.) This method also has some flexibility; It would be trivial to change the weighting of each rating so that, for example, an extremely high rating or an extremely low one would have a lessened effect on the overall score of the film.

**Possible Disadvantages to This Approach:**
As one might notice when running the *MovieExplore data/ top* command, the scores of movies drop off rather rapidly. When using the data set provided, the movie in 5th place has a score of only 70/100. While this does not affect the relative ratings of movies (the 3rd place movie is still more popular than the 100th place movie), the usefulness of the percentage scores by themselves is diminished. Someone might not be inclined to see a movie that only scored 70/10 if they didn't know that it was the 5th most popular movie in the entire database. This could possibly be solved, or at least improved, by changing the weightings of each class of rating and/or applying some sort of close-to-normal distribution to the scores. Another thing to note about this approach is that the highest scoring movie in the database will always receive a score of 100 (even if it was rated 1/1 by everyone). This isn't necessarily a drawback, but it is worth mentioning.