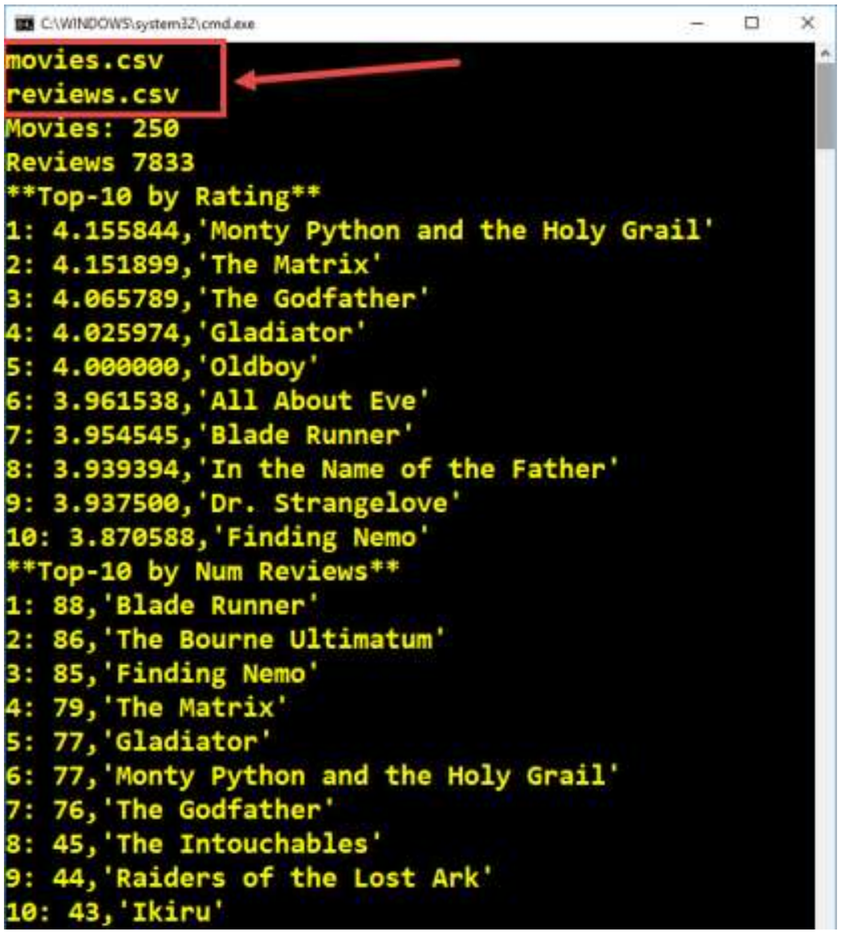
Project #01: Netflix Movie Review Analysis

Complete By: Saturday, January 14th @ 11:59pm
Assignment: C program to perform movie review analysis
Policy: Individual work only, late work *is* accepted (see "Policy" section on last page for more details)
Submission: online via zyLabs ("Project01 Netflix Analysis", section 1.15)

Assignment

The assignment is to input Netflix movie data, perform some simple analyses, and output the top-10 movies in terms of (a) average rating, and (b) number of reviews. The data will come from 2 input files, both in CSV format (Comma-Separated Values). Your job is to write a C program to input this data, organize it into a data structure, and perform the requested analyses / output.

Here's a screenshot for the input files "movies.csv" and "reviews.csv", which are supplied on the course web page. The user inputs the filenames (movies file first), and then the program inputs the data, performs the analyses, and outputs the results as shown. Your program must match this output exactly, since it will be graded using the zyLabs system.



```
movies.csv
reviews.csv
Movies: 250
Reviews 7833
**Top-10 by Rating**
1: 4.155844,'Monty Python and the Holy Grail'
2: 4.151899,'The Matrix'
3: 4.065789,'The Godfather'
4: 4.025974,'Gladiator'
5: 4.000000,'Oldboy'
6: 3.961538,'All About Eve'
7: 3.954545,'Blade Runner'
8: 3.939394,'In the Name of the Father'
9: 3.937500,'Dr. Strangelove'
10: 3.870588,'Finding Nemo'
**Top-10 by Num Reviews**
1: 88,'Blade Runner'
2: 86,'The Bourne Ultimatum'
3: 85,'Finding Nemo'
4: 79,'The Matrix'
5: 77,'Gladiator'
6: 77,'Monty Python and the Holy Grail'
7: 76,'The Godfather'
8: 45,'The Intouchables'
9: 44,'Raiders of the Lost Ark'
10: 43,'Ikiru'
```

Input Files

The input files are text files in **CSV** format — i.e. comma-separated values. The first file defines the set of **movies**, in no particular order (could be sorted by movie name, could be sorted by publication year, or could be entirely random). An example of the format is as follows:

```
250
7,12 Angry Men,1957
95,2001: A Space Odyssey,1968
211,8 and a Half,1963
 .
 .
 .
129,Wild Strawberries,1957
100,Witness for the Prosecution,1957
107,Yojimbo,1961
```

The first line is an integer N that defines how many movies are in the file; you may assume N >= 10, but do not assume a maximum size. The data starts on line 2, and each data line contains 3 values: a movie ID (integer), the movie name (string, at most 255 characters), and the year the movie was published (integer). Note that this is just one possible input file, we will use different files when grading (with a different # of movies and different movie data — but the file format will remain the same).

The second input file defines the set of movie **reviews**, in no particular order. An example of the format is as follows:

```
7833
241,1488844,3,2005-09-06
51,822109,5,2005-05-13
248,885013,4,2005-10-19
 .
 .
 .
16,55205,2,2004-10-06
34,1941094,4,2004-10-06
236,1264380,3,2004-10-07
```

The first line is an integer R that defines how many reviews are in the file; assume R > 0. The data starts on line 2, and each data line contains 4 values: the movie ID for this review, the user id (integer), the rating (an integer value in the range 1..5, inclusive), and the review date. Note that this is just one possible input file, we may use a different file when grading; the user id and review date are not used in this assignment.

The input files are available on the course web page under "Projects", then "project01-files". The files are being made available for each platform (Linux, Mac, and PC) so you may work on your platform of choice. Even though the files end in the file extension ".csv", these are text files that can be opened in a text editor (e.g. Notepad) or a spreadsheet program (e.g. Excel). **Note**: the best way to download the files is *not* to click on them directly in dropbox, but instead use the "download" button in the upper-right corner of the dropbox web page.

Requirements

In CS 251, how you solve the problem becomes as important as getting the correct answers. To encourage good programming practices, and use of appropriate data structures, your solution must meet the following requirements:

- Dynamically allocated array, or linked-list, of structs. By "dynamically-allocated", this means dynamically react to the # of movies N on the first line of the movies input file. For example, if you plan to use an array, then use N to call malloc & allocate the exact amount of memory you need. Use the same technique if you plan to store the reviews in a data structure.
- Use C; C++ constructs and containers are not allowed.
- To compute the top-10, sort. You can use whatever algorithm you want, since the # of movies is relatively small. However, you must write your own sort function, you cannot use a built-in or 3rd-party library. Note that the sort will need to be a two-level sort: e.g. you sort by average rating, but then if two movies have the same rating, you then sort by movie name. Likewise for sorting by # of reviews (e.g. see screenshot on page 1, there are 2 movies both with 77 reviews).
- No global variables whatsoever; use functions and parameter passing.

We are serious about these requirements. Use global variables instead of passing parameters? Score of 0. Use the built-in sort? 0.

Programming Environment

You are welcome to program on whatever platform you want, using whatever compiler / programming environment you want; common options were presented in the syllabus. To facilitate testing, we'll be using the zyLabs system for submission and grading; see "**Project01 Netflix Analysis**", section 1.15. You can work entirely within zyLabs in "Develop" mode, or you can download the input files from the course web page (along with a sample "main.c" to get you started) and work outside zyLabs.

When you are ready to submit your program, copy-paste your program if necessary, switch to "Submit" mode, and click SUBMIT FOR GRADING.

Getting Started

The best advice I can give is to start slow — never try to write the complete program in one sitting. Instead, like any engineering project, build it one step at a time, moving from a working program to a working program. Professionals work this way, and after 30+ years I continue to work this way. Here's a suggested set of steps:

1. The input files are CSV files. Solve the zyLabs exercise "Parsing CSV" before you start here.
2. Input the movies and output say the first 3 and last 3 stations; confirm the output matches

the first and last 3 stations in the file. [*Hint: on linux, use "more" and "tail" commands to see the beginning and end of a file.*] A common error is missing the last line, or inputting the last line twice. Tip: different platforms have different end-of-line (EOL) characters. It's a mistake to assume lines always end in \n. Instead, when inputting an entire line, I find it a good practice to input the data using **fgets**, and then stripping the EOL characters from the input in a platform-neutral way:

```
char  line[256]
int   linesize = sizeof(line) / sizeof(line[0]);

fgets(line, linesize, stdin);  // input a line from the keyboard:

line[strcspn(line, "\r\n")] = '\0';  // strip EOL(s) char at end:
```

3. I like to write functions that perform the input task. If you are working with dynamically-allocated arrays, this implies the function has to return 2 things: (1) a pointer to the start of the array, and (2) the # of elements in the array. My design has the function return the pointer to the start of the array, and then "return" the # of elements via a parameter:

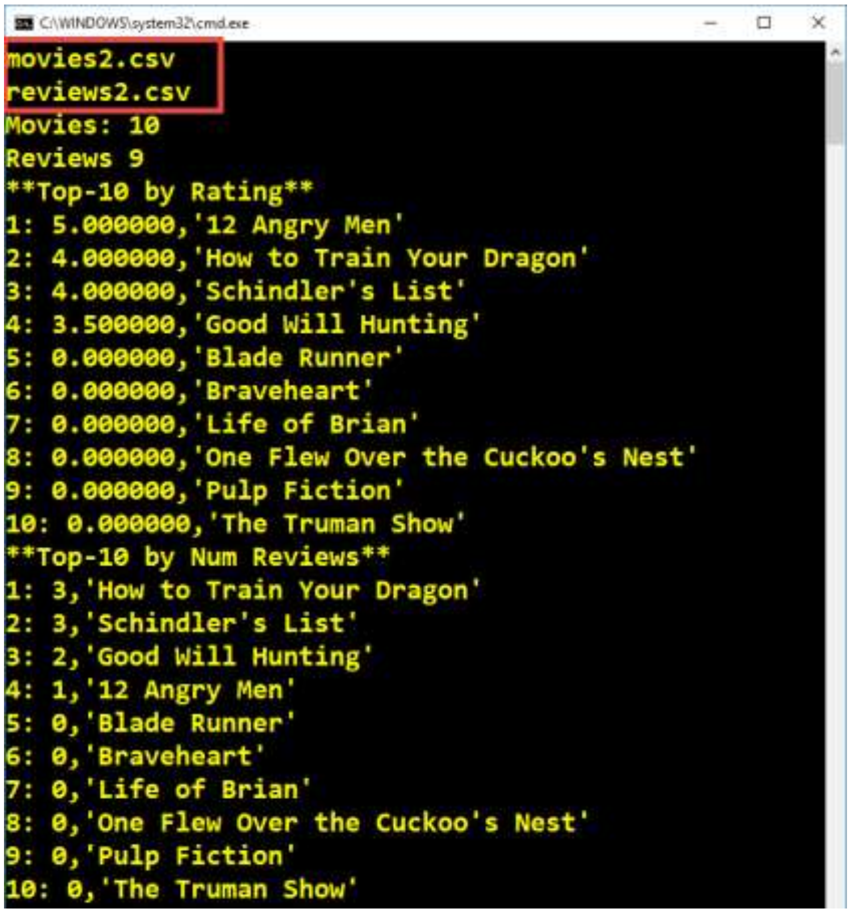
```
Movie *InputMovies(char *filename, int *numMoviesPtr)
{
  .
  .
  .

  // done, return:
  *numMoviesPtr = N;
  return movies;
}
```

4. If you haven't already, store the movie data into your data structure, and confirm that at least the first 3 and last 3 movies are stored correctly.
5. Input the reviews, and output the first and last 3 sets of data; confirm output matches contents of file. Store the review data into a data structure if you want; I chose not to.
6. Now start processing the reviews. For each review, update the corresponding movie data. Would sorting the movie data make the updating process more efficient?
7. Sort by average rating (primary sort) and movie name (secondary sort). Output top-10. Does the output match?
8. Repeat for number of reviews.

<< continued on next page >>

9. Here's another test case:



```
movies2.csv
reviews2.csv
Movies: 10
Reviews 9
**Top-10 by Rating**
1: 5.000000,'12 Angry Men'
2: 4.000000,'How to Train Your Dragon'
3: 4.000000,'Schindler's List'
4: 3.500000,'Good Will Hunting'
5: 0.000000,'Blade Runner'
6: 0.000000,'Braveheart'
7: 0.000000,'Life of Brian'
8: 0.000000,'One Flew Over the Cuckoo's Nest'
9: 0.000000,'Pulp Fiction'
10: 0.000000,'The Truman Show'
**Top-10 by Num Reviews**
1: 3,'How to Train Your Dragon'
2: 3,'Schindler's List'
3: 2,'Good Will Hunting'
4: 1,'12 Angry Men'
5: 0,'Blade Runner'
6: 0,'Braveheart'
7: 0,'Life of Brian'
8: 0,'One Flew Over the Cuckoo's Nest'
9: 0,'Pulp Fiction'
10: 0,'The Truman Show'
```

Have a question? Use Piazza, not email

As discussed in the syllabus, questions must be posted to our course Piazza site — questions via email will be ignored. Remember the guidelines for using Piazza:

1. *Look before you post* — the main advantage of Piazza is that common questions are already answered, so search for an existing answer before you post a question. Posts are categorized to help you search, e.g. "Pre-class" or "HW".

2. Post publicly — only post privately when asked by the staff, or when it's absolutely necessary (e.g. the question is of a personal nature). Private posts defeat the purpose of piazza, which is answering questions to the benefit of everyone.

3. Ask pointed questions — do not post a big chunk of code and then ask "help, please fix this". Staff and other students are willing to help, but we aren't going to type in that chunk of code to find the error. You need to narrow down the problem, and ask a pointed question, e.g. "on the 3rd line I get this error, I don't understand what that means…".

4. Post a screenshot — sometimes a picture captures the essence of your question better than text. Piazza allows the posting of images, so don't hesitate to take a screenshot and post; see http://www.take-a-screenshot.org/ .

5. Don't post your entire answer — if you do, you just gave away the answer to the ENTIRE CLASS. Sometimes you will need to post code when asking a question --- in that case post only the fragment

that denotes your question, and omit whatever details you can. If you must post the entire code, then do so privately --- there's an option to create a private post ("visible to staff only").

Submission

The program is to be submitted via zyLabs: ; see "**Project01 Netflix Analysis**", section 1.15. The grading scheme at this point is 90% correctness, 10% style — we expect basic commenting, indentation, and readability. You should also be using functions, parameter passing, and good naming conventions.

Policy

Late work *is* accepted. You may submit as late as 24 hours after the deadline for a penalty of 25%. After 24 hours, no submissions will be accepted.

All work is to be done individually — group work is not allowed. While I encourage you to talk to your peers and learn from them (e.g. via Piazza), this interaction must be superficial with regards to all work submitted for grading. This means you *cannot* work in teams, you cannot work side-by-side, you cannot submit someone else's work (partial or complete) as your own. The University's policy is described here:

http://www.uic.edu/depts/dos/docs/Student%20Disciplinary%20Policy.pdf .

In particular, note that you are guilty of academic dishonesty if you extend or receive any kind of unauthorized assistance. Absolutely no transfer of program code between students is permitted (paper or electronic), and you may not solicit code from family, friends, or online forums. Other examples of academic dishonesty include emailing your program to another student, copying-pasting code from the internet, working in a group on a homework assignment, and allowing a tutor, TA, or another individual to write an answer for you. Academic dishonesty is unacceptable, and penalties range from failure to expulsion from the university; cases are handled via the official student conduct process described at http://www.uic.edu/depts/dos/studentconductprocess.shtml .