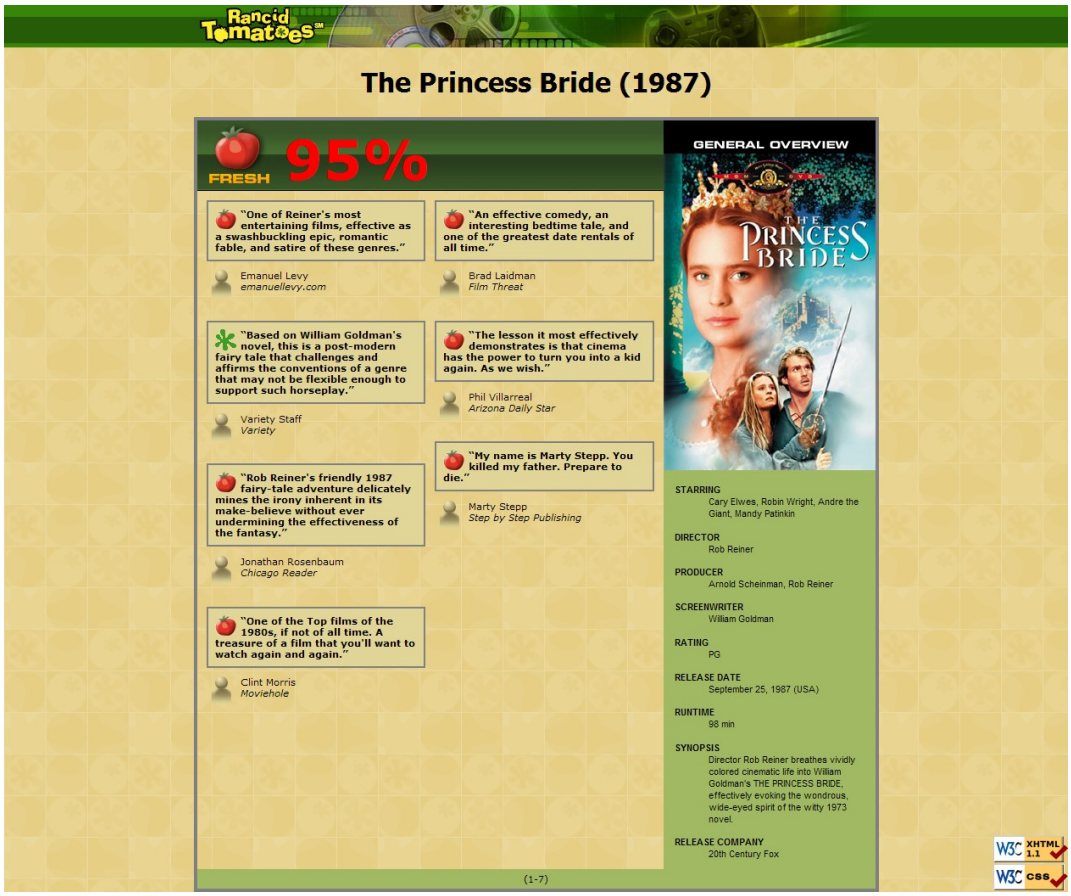
# University of Washington, CSE 190 M
# Homework Assignment 3: Movie Review Part Deux

For this assignment you will write PHP code for movie review pages much like your TMNT page from HW2. Your PHP code will allow you to generate reviews for a variety of movies using the same code. Turn in these files:

- `movie.php`, the PHP code to produce review pages for movies
- `movie.css`, the style sheet for `movie.php` *(does not need to be modified from HW2 version)*

We do not expect you to produce a pixel-perfect page that exactly matches our images. But your page should follow the styles specified in this document and match the look, layout, and behavior shown here as closely as possible.

The overall page looks the same as it did in the previous assignment. The difference is that the page might show reviews and information for a film other than TMNT. For example, the following output shows The Princess Bride:




To make things simpler, the page's title should now always be just **"Rancid Tomatoes"**.

From the course web site you will download a **.ZIP archive** of input files for many movies, described on the next page. Unzip this file into the same directory as your HW3 files, so that the files will be located in relative paths such as **tmnt2/info.txt** and **princessbride/overview.png**. Your code should assume these are the relative paths to use.

The page uses the same **images** as HW2, with two minor changes. The **tmnt.html** page showed **rottenbig.png** next to its 33% rating. But now some films have high overall ratings. Any film whose rating is 60% or above should show **freshbig.png**, shown at right. Also **overview.png** has moved, since each movie has its own overview image. Link to all images using their full **absolute URLs**, not relative ones. The overview images are the one exception; since these now reside in the folder with the movie's information, you should link to these using a relative path such as `"tmnt2/overview.png"`.


All style elements not described here should be as specified in HW2 or subject to the preference of the web browser. The screenshots in this document were taken on Windows XP in Firefox 3.5, which may differ from your system.

**Appearance Details:**

Your **movie.php** page will show reviews of different movies based upon a *query parameter* named `film` that is passed from the browser to the page in its URL. The value of this variable should be a string representing the movie to display. The browser will request your page with a URL such as the following:

**https://webster.cs.washington.edu/*your_uwnetid*/hw3/movie.php?film=princessbride**

Your PHP code can store this parameter's into a variable using code such as the following:

```
$movie = $_REQUEST["film"];
```

All of your PHP code should use this parameter's value, and you should never hard-code particular movie names. Your code may assume that the browser has properly supplied this parameter and has given it a valid value. You may assume that the movie exists and has a corresponding folder of valid input data and images. You do not have to handle the case of a missing or empty film parameter, a value that contains bad characters, a value of a movie that is not found, etc.

Based upon the movie name variable's value, display a review of that film. Each film is stored in a directory named the same as your query parameter. For example, the film `princessbride` stores its files in a folder named **princessbride/** . The files associated with each movie are the following. You may assume that the files exist and are valid at all times.

- **info.txt**, a file with three lines of information about the film: its **title**, **year**, and **overall rating** percentage. The information is used as the page heading and overall red movie rating number. Here's an example file:

```
The Princess Bride
1987
95
```

- **overview.txt**, a file with information to be placed in the General Overview section of your page. Each pair of neighboring lines contains the title and value for one item of information, to be displayed as a definition list term (`dt`) and its description (`dd`). The number of lines in the file varies from movie to movie. Example:

```
STARRING:Cary Elwes, Robin Wright, Andre the Giant, Mandy Patinkin
DIRECTOR:Rob Reiner
PRODUCER:Arnold Scheinman, Rob Reiner
SCREENWRITER:William Goldman
RATING:PG
...
```

- **overview.png**, an image to display at the top of the General Overview section. This is the movie poster for the film, such as the one shown at right.


- **review1.txt**, **review2.txt**, ..., files containing information for each review of the film. Each review file contains exactly four lines: The review, a fresh/rotten rating, the reviewer's name, and the publication. For example, `review1.txt` might store the following:

```
One of Reiner's most entertaining films, effective as a ...
FRESH
Emanuel Levy
emanuellevy.com
```

  - Different movies have different **numbers of reviews**. Show the first half of the reviews in the left column, and the rest on the right. If a movie has an odd number of reviews, the left column receives the extra review. For example, Princess Bride has 7 reviews, with the first four go on the left and the last three on the right. If a movie has only a single review, it goes in the left column. Do not worry about the possibility that the overview and reviews sections will be wildly different in height. *(You may assume that the film has at least 1 review, but it might have 10 or more; you should display them all.)*

  - The **image to show for each review** is affected by the second line of the review's text file. If the critic gave the film a FRESH rating, display **fresh.gif**. If it's ROTTEN, display **rotten.gif**.

  - If a review has 10 or more reviews (such as `tmnt2`), the names will be e.g. **review01.txt**, .... Don't hard-code file names like `"review1.txt"`; look for all files that begin with `"review"` and end with `".txt"`.

For example, if your variable stores `"princessbride"`, open **princessbride/info.txt** to read the film's title/year/etc. and display that on the page. Open **princessbride/overview.txt** and display its contents in the General Overview section on the right side. Then look for all review files in `princessbride/` and display each in the reviews area.

### Creative Aspect: Your Own Movie

As part of your turnin, create your own set of movie input data to be fed into your **movie.php** page. Write an **info.txt**, **overview.txt**, and at least 4 review text files for your movie. Also find a suitable image to use as **overview.png** of size 250x412px. You can resize an image using the free GIMP editor or a program of your choice. Turn in your movie's files as **mymovie.zip** along with your assignment.

### Development Strategy and Hints:

PHP code is difficult to **debug** if you have errors. Write the program **incrementally**, adding small pieces of code to a working page, and not advancing until you have tested the new code. We suggest that get your page to display a single film (such as `princessbride` or `tmnt2`) first, then generalize it for other films. None of your code should refer to specific names of films such as `princessbride`. The following functions may be helpful:

- `count` - returns the number of elements in an array
- `explode` - breaks apart a string into an array of smaller strings based on a delimiter
- `file` - reads the lines of a file and returns them as an array
- `glob` - given a file path or wildcard such as `"foo/bar*.jpg"`, returns an array of matching file names
- `list` - unpacks an array into a set of variables; useful for dealing with fixed-size arrays of data
- `trim` - removes whitespace at the start/end of a string (gets rid of stray spaces in output)

### Implementation and Grading:

Still turn in your **movie.css** even if it did not change. Our **movie.php** solution is about 115 lines long (85 "substantive").

For full credit, your page's output for all films must successfully pass the W3C **XHTML validator**. (Not the PHP source code itself, but the HTML output it generates, must validate.) Do not use HTML tables on this assignment.

Your PHP code should not cause errors or warnings. We will also grade the **style of your PHP code**; like a CSE 14x assignment. Minimize use of the `global` keyword, use indentation/spacing, and avoid long lines over 100 characters. Use material from the first three weeks of class and the first five book chapters. (Book Chapter 5 is a good resource.)

Be mindful of "mode switches" between PHP code and HTML tags using `<?php` and `?>` . Produce as much of your output in HTML mode as possible; only switch to "PHP mode" to compute or print dynamic content. For full credit, you should reduce the number of large complex chunks of PHP code that are placed in the middle of your HTML code. When possible, replace such chunks with functions that are called in the HTML and declared at the bottom of your file.

Also, you should minimize the use of `print` and `echo` statements. As much as possible you should insert dynamic content into the page using PHP expression blocks as taught in class. You will lose points if you print any HTML tags into the page such as the following:

```
print "<p>I am adding content to the page! $x $y $z</p>";    # bad
```

Another grading focus is **redundancy**. You should not have code that depends on particular movies or uses `if/else` statements to see which movie to display. Use loops, variables, `if/else` factoring, etc. to avoid redundancy.

Another grading focus is PHP **commenting**. We expect more comments here, similar to CSE 14x. Put a descriptive header (name, course, TA, description) at the top of your code and comment each section of PHP code.

**Format your HTML and PHP code** similarly to the examples from class. Properly use whitespace and indentation. Do not place more than one block element on a line or begin a block element past the 100th character. **CSS** is not a major part of this assignment, but you should not introduce new poorly written CSS code or styles.

Part of your grade will also come from successfully uploading your files to the **Webster** web server in a subdirectory named `hw3`, so that it is possible to navigate to your page by entering the following URL into the browser:

- **https://webster.cs.washington.edu/*your_uwnetid*/hw3/movie.php**

Please do not place a solution to this assignment online on a publicly accessible web site.