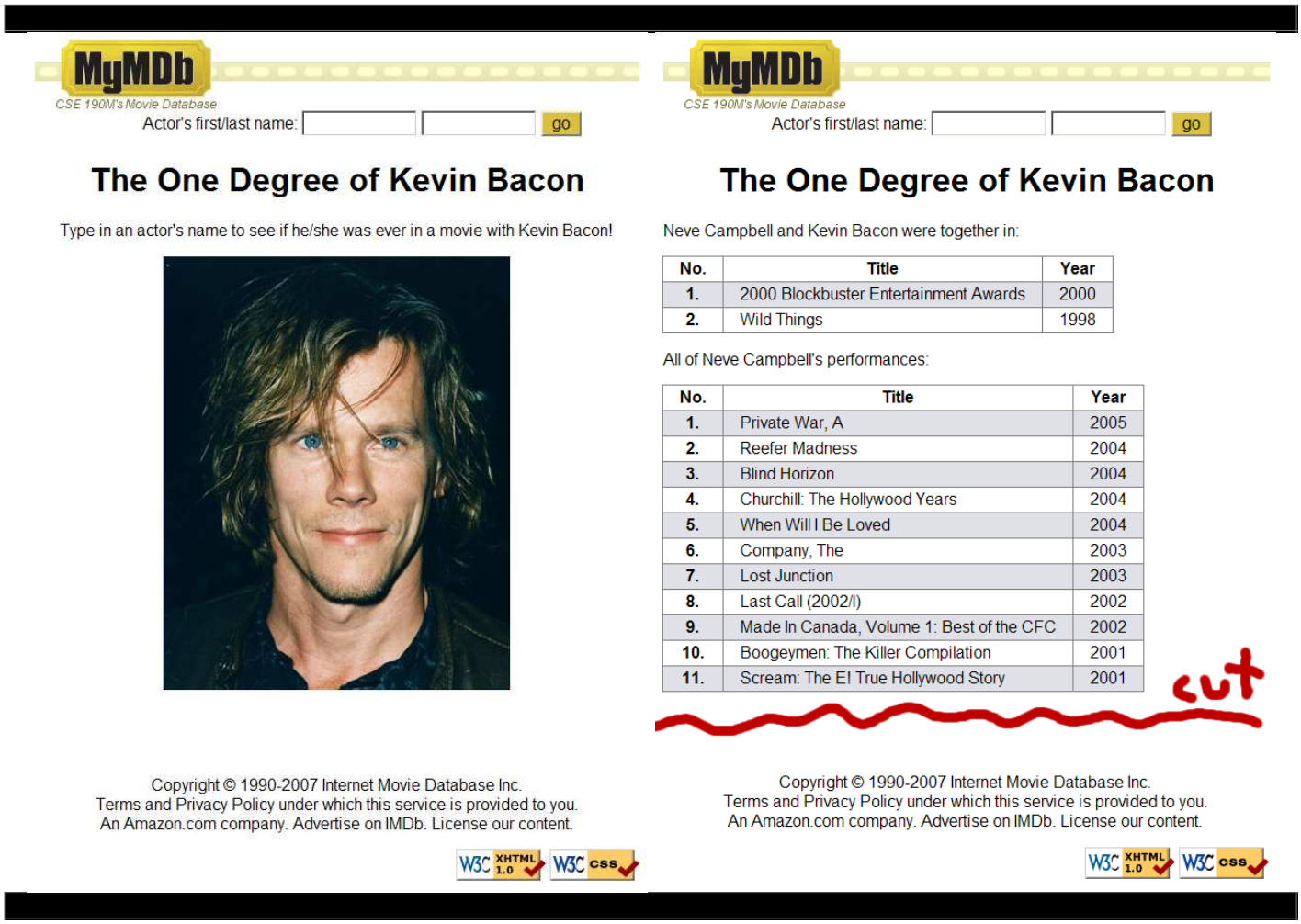
University of Washington, CSE 190 M, Spring 2007
Homework Assignment 8: Kevin Bacon

due Thursday, May 31, 2007, 11:59pm electronically

This assignment asks you to use PHP and SQL. You will write the following page:



Your task for this assignment is to write the HTML and PHP code for a web site called *MyMDb* that mimics some of the functionality of the popular IMDb movie database site. The main ability of your site will be to link other actors to Kevin Bacon. If you prefer, you may use another actor of your choice rather than Bacon as the center point. The site will also let the user search for an actor by an approximate name, as well as displaying all movies by a given actor.

The site will consist of pages called `index.php` and `results.php`. The `index.php` page contains some initial content of your choice (described later) along with a form for a user to search for information about an actor. When the user submits a search, the browser is redirected to `results.php`, which displays the search results.

Turn in the two `.php` files mentioned previously, along with any other files you create. You should have a `.css` stylesheet file for any stylistic information. You should also identify common pieces of text or code and express these as separate files that are included in your pages.

Part of your grade will come from posting your program to *exactly* the following URL:

`https://students.washington.edu/`***`your_UWnetID`***`/cse190m/homework/8/`

Page Appearance - Both Pages:

The images referred to in this document exist in the following folder:

`http://www.cs.washington.edu/education/courses/cse190m/CurrentQtr/homework/8-mymdb/images/`

All pages have a title of `The CSE 190M Internet Movie Database (MyMDb)` and a "favorites icon" of `mymdb_icon.gif`. The font of all text on all pages is 12pt Arial, or the default sans-serif font on the system.

All pages begin with the MyMDB logo image of `mymdb.png`. Spanning behind this is a horizontally repeating image of `background.png`. Below this is a form where the user can type an actor's first/last name, with a `go` button (colored `DAC23D`) that submits the form data to `results.php`. You may use either a GET or POST request as you see fit.

Below the form is a central area in which results are displayed. In `index.php` this area contains custom content of your own. You must include at least one image of Kevin Bacon (or the actor of your choice) and text explaining the program. Put as much content here as you like, even multimedia or links; be creative!

5em below this, the bottom of the page includes a paragraph of copyright information, followed by the standard W3C validator buttons in the right edge. The copyright information is:

> Copyright © 1990-2007 Internet Movie Database Inc.
> Terms and Privacy Policy under which this service is provided to you.
> An Amazon.com company. Advertise on IMDb. License our content.

Page Appearance - `results.php`:

The central content area of the `results.php` page shows various content dependent on the query the user submitted. There are several cases that your code must handle:

Blank name:

Your page should handle specially the case where both the first and last name are blank. **Your code should not display any data nor perform any database queries if both names are blank.** This is done to save wear and tear on the poor database server, which might otherwise return ALL actors from such a case.

(a) Name doesn't match any actor in the database:

Your central content area should contain a paragraph with the following message, where [NAME] is the name the user typed (first name, space, last name):

> Sorry, there were no actors matching [NAME].

You can determine how many actors match a given name by seeing how many rows were returned from your query. Do this by calling the `mysql_num_rows` function as described in the SQL lecture slides.

(b) Name matches multiple actors in the database:

This is likely to happen for many actor names, such as partial names, names with only a first or last name entered, or common names (such as Will Smith). In such a case, your page should display a heading saying "Actors matching [NAME]". Under this heading should be an list of actors in the database that matched the submitted name text, sorted by last name and then by first name. Show any names in the database in which the user's first and last name text matched as a substring; for example, if the user types "oh" as the first name and "sack" as the last name, the matching actors are "John Cusack" and "John Sackman".

Each of the names shown should be a link to load `results.php` with that actor's name as the query. For example, clicking "John Cusack" would reload `results.php` and show information about John Cusack.

(c) Name matches a single actor in the database:

If the user types an actor name that uniquely matches one actor in the database, you should perform two queries on the database. The first is to see in which movies, if any, that actor performed with Kevin Bacon (or the actor of your choice). Any such movies should be displayed as an HTML table. If the actor has not been in any movies with Kevin Bacon, your page should display this message instead of the table:

> [NAME] wasn't in anything with Kevin Bacon.

The second query your page should perform is to find a complete list of movies in which the actor has performed. These movies should be displayed as a second HTML table.

The data of both tables is sorted in descending order by year. Both tables have three columns: A number for each movie, starting at 1; the movie's title; and the year of release. The columns have bold centered headings. The movies' background colors alternate between `E0E1E7` (odd numbers) and white (even numbers). Cells have 1em of padding on their left and right sides, and 0.1em of padding on their top and bottom. The table and cells have a collapsed 1px solid gray border.

No.	Title	Year
1.	Private War, A	2005
2.	Reefer Madness	2004
3.	Blind Horizon	2004
4	Churchill: The Hollywood Years	2004

Screenshots in this document are from Windows XP in Firefox 2.0, which may differ from your system.

Database and Queries:

Your pages reads their data from a MySQL database named `imdb`, which is located on the server named `webster.cs.washington.edu`. Your code can query this database using PHP's MySQL functions. You will log in to this database using your UW NetID as your user name, and a password that will be emailed to you. The database contains the following tables:

```
Actor          (id, fname, lname, gender)
Cast           (aid, mid, role)
Movie          (id, name, year)
```

To solve this program, you'll need to come up with three unique SQL queries on the database. They are listed below in order from easiest to hardest. This is the order in which we recommend you develop them.

1. A query to find all the actors whose names match the first/last name the user typed.
   - Use the `LIKE` clause to search for first/last names that contain the user's query text as a substring.
2. A query to find all movies in which an actor has performed. To find this information, you will need to use a join between several tables of the database. The Cast table provides the link between a given actor's ID and the IDs of the movies he/she has been in. The Movie table gives you the other information about a movie, given its ID. You should acquire all of this information with a single SQL query; it is not acceptable to perform many queries (such as one per movie) to do so.
   - Use a `FROM` clause with several tables listed, or a `JOIN` clause, to perform a join in your query. Use `ORDER BY` to sort your results.

     Hint: You will need to join the Actor, Cast, and Movie tables, and only retain rows where the various IDs from the tables (actor ID, movie ID) match each other and the ID of your actor.

Our query names 3 tables in the `FROM` clause and 3 `WHERE` conditions separated by `AND`.

3. A query to find all movies in which both your actor and Kevin Bacon have performed together. As with the preceding query, this one involves join operations. This query involves more joining, because you must link not only your actor's ID to the movie, but also Kevin Bacon's.
    - Hint: You will need to join a pair of Actors (yours and Kevin Bacon), a corresponding pair of Cast records, and the related Movie. Only retain rows where the two Actors match the Cast records, where the Cast record matches the Movie, and where the two Cast records are for the same movie.

      Our query names 5 tables in the `FROM` clause and 7 `WHERE` conditions separated by `AND`.

Your queries should be constructed so that they effectively filter the data down to only results that are relevant. **It is not acceptable to perform a query that returns too many results, then trim it down in your PHP code.** For example, an incorrect way to find all movies the actor has been in with Kevin Bacon would be to get all of the actor's movies with one query, get all of Kevin Bacon's movies with another query, and then use PHP code to merge the two. This is unacceptable because it is taxing on the database server.

Small Database, SSH, and Wall of Shame:

Because the overall imdb database is so large, it can be very costly to perform a malformed query on it. This database will be used by all students, so a poorly written program can affect performance for everyone.

To address this problem, we have created a database named `imdb_small` that contains a smaller set of records. While you are debugging your code, please point it at that database rather than at the full `imdb` database. When you are reasonably sure that you have your program working, then you can switch to the full `imdb` dataset.

To discourage students from crashing or overloading the database server, we will post a "wall of shame" on the course web site. This wall will list the partial names of any students who break the server. (This is intended to be light-hearted; no actual punishment will occur for such mistakes.)

We also encourage you to test your queries on MySQL directly before running them as PHP code. To do this, connect to `webster.cs.washington.edu` using SSH, and then type the command `mysql -u` *`YOUR_USER_NAME`* `-p` and press Enter. It will ask for your SQL password as sent to you by email. Once you're in the MySQL prompt, type `USE imdb;` or `USE imdb_small;` to connect to the database, and then issue queries as normal. Press Ctrl-C to cancel a query in progress, and type `quit` to exit MySQL.

Submission and Grading:

Submit your assignment online from the turnin area of the course web site. For reference, our solution contains 124 unique lines of PHP code.

Your HTML code (including HTML code produced as output from PHP) should follow valid XHTML syntax and should pass the W3C XHTML validator. You should also not place stylistic information in your HTML code when it could instead be placed into a CSS stylesheet.

Your PHP code should follow reasonable stylistic guidelines similar to those you would follow on a CSE 14x programming assignment. In particular, you should minimize redundancy, follow proper procedural decomposition, correctly use indentation and spacing, and place a comment header at the top of your file and atop every function explaining that function's behavior. You should make a particularly strong effort to remove redundant HTML and PHP code from your program.

Please do not place a solution to this assignment online on a publicly accessible (un-passworded) web site.