Lab 2 Check-Off

Student Name: _____

Date of Lab: _____

Check-Offs

_____ Home page created and saved to git

_____ Theater page created and saved to git

_____ Report form created and saved to git

TA (signed): _____

TA (print): _____

Lab 2 Instructions

Lab Objectives: The primary objective of this lab to help students learn how to build a simple static website using Twitter's Bootstrap CSS. The secondary objectives are to give students additional practice with both html, css and git.

1. We will be using Twitter's Bootstrap CSS to build this static site, so it may be beneficial to go to the project's home page at http://twitter.github.com/bootstrap/ in case you have questions or need clarification. Normally we would have you clone the full repository by going to the command line and typing:

   ```
   git clone https://github.com/twitter/bootstrap.git
   ```

   but in this case we are going to only use a limited set of Bootstrap and in a limited time frame, we will work with the bgr_project_lab2.zip file contents found on the course site. Get this file and unzip it now to get these contents.

2. Now open the home.html file in a text editor and at the top of the page add a meta tag indicating that you are the author of this project. After that, add the bootstrap.css (this is from Twitter) and the extensions.css files (which is from Prof. H). In case you can't remember the exact notation, it is as follows:

   ```
   <link rel="stylesheet" href="css/bootstrap.css" type="text/css">
   <link rel="stylesheet" href="css/extensions.css" type="text/css">
   ```

   Now create a git repository and add all these files to the repo. Again, in case you forgot from last week, you can do this with the following commands on the command line (assuming your in the right directory):

   ```
   $ git init
   $ git add .
   $ git commit –m "Added head materials to home.html"
   ```

3. We are going to start with the body (the part the user sees directly in the browser) by building a simple masthead and

navigation bar. I have already made some modifications to the stylesheets such that the top_bar is relatively, not absolutely, positioned (so we can use it as the menu bar) and set up a nice gold banner as the masthead background. To position our content properly with bootstrap, we want to place our content within its default 940px container (see http://twitter.github.com/bootstrap/#layouts). Knowing that, we can build the masthead with the following:

```
<div class="masthead">
  <div class="container">
    <img src="images/bgr_logo_text.png" />
  </div>
</div>
```

This creates a div of class 'masthead' (could have used an id here as well) and within it created the container. Now the image will sit in that container and adjust as the size of the browser window changes. (Try it and see…) Once you see it working, add the home.html to staging and commit to the git repo.

4. We will do something similar for the navigation bar. We will use bootstrap's 'topbar' to serve as our nav bar in this case. The code below will suffice to create the

```
<div class="topbar">
  <div class="topbar-inner">
    <div class="container">

      <!-- links go here -->

    </div>
  </div>
</div>
```

Now to add links, create an unordered list in topbar with the class of "nav" and then add each link as a link item, being sure to wrap the actual link text in an anchor tag. If you need help, the file 'home_page_materials.txt' has what the links should look like. We will mark the home link's <li> tag as class="active" to get a little extra color on the link and help users get a signal where they are on the site. Once working, commit changes to the git repo.

5. Moving to the content section, we need to use bootstrap's build-in grid system to section off the content into two subsections; see http://twitter.github.com/bootstrap/#grid-system before proceeding. The first section in our simple grid will be 2/3 of the page and hold the main content while the other will be 1/3 of the page and have a special side 'bucket' where we will list movies made in the 'Burgh. To create the basic structure (again, remember we want this in the container div), we can do the following:

```
<div class="row">
  <div class="span-two-thirds">

     <!-- welcome content goes here -->

  </div>
  <div class="span-one-third">

     <!-- side bucket content goes here -->
     <!-- note: add all this content in a div called "bucket" -->

  </div>
</div>
```

   Create this basic structure and then add in the content provided. Review it in the browser and then commit to the git repository.

6. For the bottom of the home page create a footer (again keeping its contents within bootstrap's "container") with the following structure and add in the content provided:

```
<footer class="footer">
  <div class="container">

     <!-- footer content goes here -->

  </div>
</footer>
```

7. Before we leave the home page, let's use bootstrap's inline labels to spruce up the side bucket list. (See more information on these labels at http://twitter.github.com/bootstrap/#typography) Since the *Dark Knight Rises* isn't technically out yet, add an "upcoming" label by placing the following code next to that item in the list: `<span class="label success">Upcoming</span>` Since Smart People, Wonder Boys, and The Mothman Prophecies were all filmed at CMU and that is important to note, mark it as such with by adding: `<span`

`class="label important">CMU</span>` Confirm these changes in the browser and commit to git. (You did do that in step 6 as well, right?)

8. Moving onto the theaters.html page, our goal here is to simply add a table using bootstrap. (see http://twitter.github.com/bootstrap/#tables) Replace the comment on line 47 with an html table with a class of `"zebra-striped span10"`. Create a `<thead>` with two columns: `<th width="100">Movie</th>` and `<th width="250">Locations</th>`. After that, add in the content of the four rows (see "theater_page_materials.txt" for content) as you would an html table. Verify in the browser that it is working and commit to the git repository.

9. Moving onto the report.html page, our goal here is to simply add a form using bootstrap. (see http://twitter.github.com/bootstrap/#forms) Create an html form with method="post" (ignore the "action" attribute as this form will not actually submit). Within the form, create a set of fieldset tags and within those create a legend by adding `<legend>Tell Us What You Think</legend>`. After that, paste in the form code provided in report_page_materials.txt. View the page in the browser. It looks ugly; this is how bootstrap does it? Apparently not, but how do we fix it?

   The first thing bootstrap expects is that you will wrap all of your form input elements in div tags with class="input". For each form input type (the set of checkboxes count as one input in this case) and the submit buttons, wrap `<div class="input"> … </div>` around it; for checkboxes be sure that `<ul class="inputs-list">`. Look again in the browser. Better, but still not very good.

   The second thing bootstrap expects is that each label *and* form input together as one will be wrapped in a div of class="clearfix". (See the article http://thinkvitamin.com/design/everything-you-know-about-clearfix-is-wrong/ for more on clearfix.) Do this now and see the improvement. *The Tick* graphics even appear in the proper place – life is good and we are nearly finished. Before the TA signs off, what is the last thing you have to do? *(hint: starts with 'g' and ends with 'it')*

Be sure that the check-off sheet has your name clearly printed and is signed by a TA and take it with you as proof of lab completion.

On Your Own…

Later this week for the "on your own" section, you will be doing some SQL practice but not until after we begin covering SQL in class. First, get the data dictionary for the WPCC database online. This file can be viewed at or downloaded from (your preference) the course site at http://rook.hss.cmu.edu/~67272/labs/wpcc_data_dict.pdf. (This file uses the same template that is used in phase 1.) After reviewing this, log into the course site and then redirect your browser to http://rook.hss.cmu.edu/~67272/SQL_Lab.php. Complete each of the six queries in the fields provided. If you are incorrect, the results we are expecting appear along with the results your query generated. The autograder is very particular – it wants the exact wording, watching every capitalized letter and punctuation mark. If you need help with this, please see Prof. H or a TA. (Do not try to attempt these exercises unless you already know SQL or have gone through the SQL lectures; at the same time, this is good practice for the exam.)

plthl