### Rules:

- You have **60 minutes** to complete this part of the exam. You may receive a deduction if you keep working after the instructor calls for papers.
- This test is open-book/notes. You may use any paper resources other than practice exams.
- You may *not* use any computing devices, including calculators, cell phones, or music players.
- Unless otherwise indicated, your code will be graded on proper behavior/output, not on style.
- Please do not abbreviate code, such as writing ditto marks (""") or ellipses (…).
- If you enter the room, you must turn in an exam and will not be permitted to leave without doing so.
- You must show your **Student ID** to a TA or instructor for your submitted exam to be accepted.

**Good luck!**

<table>
<thead>
<tr>
<th>Problem</th>
<th>Description</th>
<th>Earned</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>HTML/CSS Tracing</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>HTML/CSS Coding</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>JavaScript/DOM</td>
<td></td>
<td>20</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>Day's Total Points</strong></td>
<td><strong>50</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem</th>
<th>Description</th>
<th>Earned</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>AJAX</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>PHP</td>
<td></td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>SQL</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>Day's Total Points</strong></td>
<td><strong>50</strong></td>
<td></td>
</tr>
</tbody>
</table>

**TOTAL Exam Total Points** 100
(This page intentionally left blank.)
4. Ajax/XML

Suppose that there is a web service named findit.php, located on your web server in the same directory as your code. This service outputs XML data describing turn-by-turn directions from a given origin to a given destination. In this problem you will write Ajax JavaScript code to contact the web service (using a GET request), examine its XML data, and display the series of directions contained in it on the page.

The XML data returned by the web service consists of an overall document element called directionlist that contains a series of one or more direction elements inside it. Each direction element has three attributes: a distance attribute containing the distance to travel before the next turn; a turn attribute indicating the direction of the turn; and a street attribute indicating the name of the street to turn at.

To indicate arrival at the destination, the last direction element in the list will not have a street attribute. The format matches the following example (though it might have more/fewer than 3 directions):

```xml
<?xml version="1.0" encoding="UTF-8"?>
<directionlist from="Collins Pub" to="Quinn's Pub">
    <direction distance="0.1" turn="left" street="Yesler Way" />
    <direction distance="0.5" turn="left" street="Broadway" />
    <direction distance="0.9" turn="right" street="E Pike St" />
    <direction distance="0.1" turn="arrive" />
</directionlist>
```

Assume the code you are writing will go into findit.js, linked from the following page:

**Find-It!**

Get directions:

From:
collins pub

To:
quinn's pub

Find-It!

1. ![image] in 0.1 miles turn left on Yesler Way.
2. ![image] in 0.5 miles turn left on Broadway.
3. ![image] in 0.9 miles turn right on E Pike St.
4. ![image] in 0.1 miles arrive at Quinn's Pub.

You should perform the search when the button with the ID of find is clicked, by initiating an AJAX request to the findit.php service and passing it two parameters—from and to—containing the values entered in the text boxes. The text boxes have IDs from and to respectively. Assume valid input to both text boxes.

Results should be injected as list items into the ol with the ID of directions. Each list item first contains an image whose source is equal to the value of the turn attribute with '.png' at the end. (For example, the right-turn image is right.png.)

For regular directions you should use the following format:

[![image] in [distance] miles turn [turn] on [street].

When indicating arrival, however, the message is slightly different:

[![image] in [distance] miles arrive at [destination].

You may assume that the XML data is valid in the format described previously, and that the .php service is reachable and no failures occur during the AJAX request. You may also assume that Prototype has been loaded prior to your script.
4. Ajax/XML (writing space)
5. PHP

Write the PHP code for a self-submitting form page with two states:

When a **parameter is not passed**, output a form to select (from a dropdown box) an elevation profile to display. The options in the dropdown should be the names of **all files in the current directory** that end in `.dat` as follows:

```
Elevate–It!

Select an elevation profile:

[Ragnar Leg 25.dat]

Elevate–It!
```

The form on this page should submit back to itself. When a **parameter called profile is passed**, load the file of the given name, and output its contents in a table. The data in the file will be in the following format:

```
Elevate–It!

Elevation data for Ragnar Leg 25.dat:

<table>
<thead>
<tr>
<th>distance</th>
<th>elevation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>7.1</td>
</tr>
<tr>
<td>49</td>
<td>7.1</td>
</tr>
<tr>
<td>68</td>
<td>7.2</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Each line contains two numbers, separated by a comma. The first number represents a distance, and the second represents an elevation, both in meters. You should output these two numbers in a table as follows:

```
At the end of your page, you should print a **cumulative gain/loss** which indicates the total amount of elevation gained or lost, as follows:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>802</td>
<td>11.4</td>
</tr>
<tr>
<td>820</td>
<td>11.6</td>
</tr>
<tr>
<td>838</td>
<td>11.8</td>
</tr>
<tr>
<td>855</td>
<td>11.9</td>
</tr>
<tr>
<td>878</td>
<td>12</td>
</tr>
</tbody>
</table>

Total gain/loss: 408.7
```

Your code doesn't need to output a complete XHTML page; assume that your output will be put inside the page's **body**. You will receive a small deduction if you use `print` or `echo` statements in your solution.
5. PHP (writing space)
6. **SQL**

Some actors are known for working with particular directors. Write a SQL query that finds a list of all actors in the *imdb* database who worked on two films with the same director.

Show the first and last name of the director, then the first and last name of the actor. Each combination of actor and director should be listed only once. Recall the *imdb* database tables:

- **actors**
  - id
  - first_name
  - last_name
  - gender
  - 433259 William Shatner M
  - 797926 Britney Spears F
  - 831289 Sigourney Weaver F

- **movies**
  - id
  - name
  - year
  - rank
  - 112290 Fight Club 1999 8.5
  - 209658 Meet the Parents 2000 7
  - 210511 Memento 2000 8.7

- **roles**
  - actor_id
  - movie_id
  - role
  - 433259 313398 Capt. James T. Kirk
  - 433259 407323 Sgt. T.J. Hooker
  - 797926 342189 Herself

- **directors**
  - id
  - first_name
  - last_name
  - 24758 David Fincher
  - 66965 Jay Roach
  - 72723 William Shatner

- **movies_directors**
  - director_id
  - movie_id
  - 24758 112290
  - 66965 209658
  - 72723 313398

- **movies_genres**
  - movie_id
  - genre
  - 209658 Comedy
  - 313398 Action
  - 313398 Sci-Fi

If you unnecessarily join too many tables together that are not needed for the query, you will not receive full credit.