CSE 154: Web Programming Autumn 2018

Practice Final Exam 1

Name:

UWNet ID: @uw.edu

TA (or section):

Rules:

- You have 120 minutes to complete this exam.
- You will receive a deduction if you keep working after the instructor calls for papers.
- You may not use any electronic or computing devices, including calculators, cell phones, smartwatches, and music players.
- Unless otherwise indicated, your code will be graded on proper behavior/output, not on style.
- Do not abbreviate code, such as writing ditto marks ("") or dot-dot-dot marks (...). You may not use JavaScript frameworks such as jQuery or Prototype when solving problems.
- 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.

Question	**Score**	**Possible**
HTML/CSS		15
Short Answer		18
JS/DOM/Animations		12
JS with AJAX		20
PHP Web Service		20
SQL		15

1. HTML/CSS (Part A): Query Selectors

Consider the following HTML:

```
<html>
    <heading>
        <title>CSE 154 Course Web Page</title>
    </heading>
    <body>
        <header id="title-1">
            <h1 id="title-2"><em id="em-1">All the CSE 154 Course Stuffff Ever</em></h1>
        </header>
        <p id="subtitle-1">Topics:</p>
        <ul id="list-1">
            <li id="topic-1">What is the Internet</li>
            <li id="topic-2">How to do the Internet</li>
            <li id="topic-3">How to make the Internet</li>
            <li id="topic-4">
                Make cool projects:
                <ol id="list-2">
                    <li id="hw-1">Make Pies</li>
                    <li id="hw-2">Watch Lion King</li>
                    <li id="hw-3">Read <em id="em-2">rly rly rly</em> fast</li>
                    <li id="hw-4">Push squares around</li>
                    <li id="hw-5">Catch 'em all!</li>
                </ol>
            </li>
        </ul>
        <div id="div-1">
            <img id="img-1" src="mowgli.jpg">Our course mascot!</img>
        </div>
    </body>
</html>
```

Write the ID's of the elements selected by each of the given selectors:

1. `p`

____

2. `ol li`

____

3. `li em`

____

4. `ul > li`

____

5. `li li`

____

1. HTML/CSS (Part B): Writing CSS

Given the following HTML body, write the CSS to fit the requirements.

```
<body>
  <h1>A Collection of the best recipes ever</h1>
  <article id="recipe-list">
    <ul>
      <li>Holiday Cookies</li>
      <li>Lasagna</li>
      <li>Ants on a Log</li>
      <li>Kimchi Burrito</li>
    </ul>
  </article>
  <article id="recipe-area">
    <h2>Holiday Cookies</h2>
    <article id="ingredients">
      <ul>
        <li>flour</li>
        <li>sugar</li>
        <li>magic</li>
      </ul>
    </article>
    <p class="instruction">Combine all the ingredients.</p>
    <p class="instruction">Wish on a shooting star.</p>
    <p class="instruction">Profit.</p>
  </article>
</body>
```

Styling Requirements:

- The background of the entire page should be #123456.
- The color of the text for all the headings should be teal.
- Every element with the class instruction should have a border that is 2 pixels, dashed, and red.
- The article with the id recipe-list width should take up 40% of its parent's width.
- The article with the id recipe-area width should take up 60% of its parent's width.
- The text in the list inside the article with the id ingredients should have a font preference of Arial, Helvetica, or any other sans-serif font.

Write your solution to Problem 1 (Part B) on the next page.

Write your solution to Problem 1 (Part B) below:

2. Short Answers

1. What is one reason to use semantic tags instead of a <div> in HTML?

---

2. What is one example of validating user input on the client-side?

---

3. What’s the difference between margin, borders, and padding? (You may provide a labeled diagram)

---

4. What is the difference between `setInterval` and `setTimeout`?

---

5. What is the difference between a GET and POST request?

---

6. What is one advantage of using a SQL database over text files to store data?

---

7. For each of the two regular expressions, circle all the string(s) below that match it:

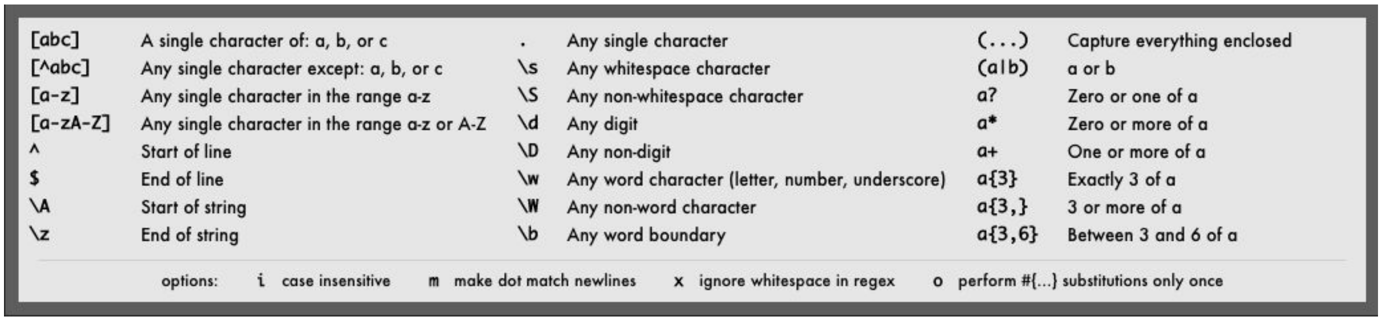
i. `/[A-Za-z]+@[0-9]/`

- `foo+@5`
- `foo@123`
- `45foo@321`
- `f8@8f`

ii. `/^F*\.jpg$/`

- `F.jpg`
- `^FFF.jpg$`
- `FFF.jpg`
- `F\.jpg`

Regex reference:



[abc]	A single character of: a, b, or c	.	Any single character	(...)	Capture everything enclosed
[^abc]	Any single character except: a, b, or c	\s	Any whitespace character	(a|b)	a or b
[a-z]	Any single character in the range a-z	\S	Any non-whitespace character	a?	Zero or one of a
[a-zA-Z]	Any single character in the range a-z or A-Z	\d	Any digit	a*	Zero or more of a
^	Start of line	\D	Any non-digit	a+	One or more of a
$	End of line	\w	Any word character (letter, number, underscore)	a{3}	Exactly 3 of a
\A	Start of string	\W	Any non-word character	a{3,}	3 or more of a
\z	End of string	\b	Any word boundary	a{3,6}	Between 3 and 6 of a

options: i case insensitive m make dot match newlines x ignore whitespace in regex o perform #{...} substitutions only once

8. Suppose a directory has the following structure:

```
test.php
mydir/
    images/
        puppy1.jpg
        puppy1.png
        puppy2.gif
    puppy-facts.txt
    puppy-haz-pizza.jpg
```

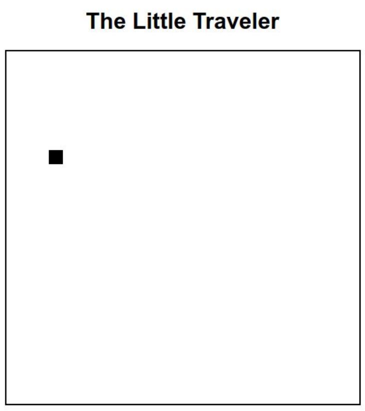
What do each of the following statements return if written in test.php? Use [] notation for any arrays and put strings in "".

Statement	**Return Value**
`scandir("mydir")`	
`scandir("mydir/images")`	
`glob("mydir/puppy-facts.txt")`	
`glob("mydir/*/*")`	
`glob("mydir/puppy*")`	

3. The Little Traveler (JS/DOM/Animations)

Given the HTML and CSS on the following page, write the JavaScript code that adds a `.little-box` div to the top left corner of the `#box` div, and moves the little box inside of the `#box` 20px up, down, left, or right randomly every 100ms. The little box may only move to a position that is inside of the boundaries of the parent `#box`. Note that the `#box` parent has a width and height of 500px, and the `.little-box` div has a width and height of 20px (without any border):

To the right is a screenshot of the little box during an animation. Write your JavaScript solution on the next page following the provided HTML and CSS:



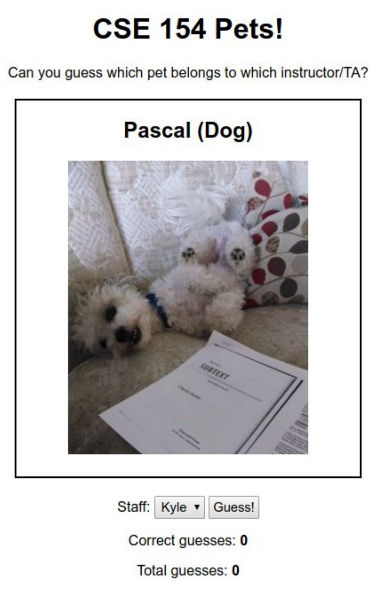
```
<!DOCTYPE html> <!-- HTML for Problem 3 -->
<html lang="en">
  <head>
    <link href="traveler.css" rel="stylesheet" />
    <script src="traveler.js"></script>
  </head>
  <body>
    <h1>The Little Traveler</h1>
    <div id="box"></div>
  </body>
</html>
```

```
/* CSS for Problem 3 */
h1 {
  font-family: Helvetica, Arial, sans-serif;
  text-align: center;
}

.little-box {
  background-color: #000;
  height: 20px;
  position: absolute;
  width: 20px;
}

#box {
  border: 2px solid black;
  height: 500px;
  margin: auto auto;
  position: relative;
  width: 500px;
}
```

Write your solution to Problem 3 below:

```
(function() {
  "use strict";

})();
```

4. Fetching Pets Who Fetch with Fetch (JS/AJAX)


CSE 154 Pets!

Can you guess which pet belongs to which instructor/TA?

Pascal (Dog)

Staff: Kyle ▾ Guess!

Correct guesses: **0**

Total guesses: **0**

Write a JavaScript file pets.js that plays a guessing game with the client, displaying the name and image of a random pet owned by a CSE 154 staff member, and which keeps track of how many times the client correctly guesses the TA/instructor who owns the pet. To the right is a screenshot of the page **before** a user has made any guesses.

Initial Behavior

When the page loads, you should make an AJAX request to pets.php with a query parameter of mode=tas. Use the returned (plain text) response to populate #ta-list with a new option tag for each TA/instructor name in the result. (Hint: remember that each option tag in a dropdown should have a unique value attribute to determine what the current selection is).

Example response from `pets.php?mode=tas`:

```
Conner
Jeremy
Kyle
Lauren
Melissa
Sam
```

In addition to populating the dropdown with staff names, you should initialize the guessing game with the first random pet. You should get this data with a call to pets.php, passing the query parameter mode=random. Using the JSON response returned, you should update #pet-name and #pet-type to be the name and type of the random pet and update the source of #pet-img to be a randomly-selected image source path contained in the JSON's images array. If the array contains only one image, you should use this image as the source. When the pet image is first changed, remove the initial "hidden" class from #pet-img (you should not hide it again).

Example response from `pets.php?mode=random`:

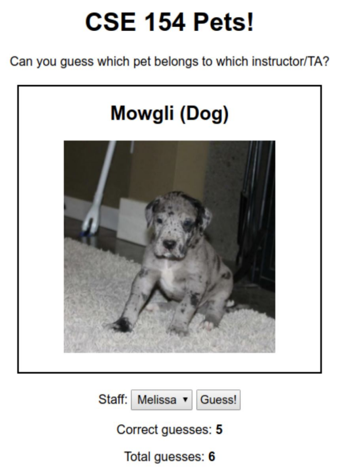
```
{
  name : "Melissa",
  petname : "Mowgli",
  type : "Dog",
  age : "10 months",
  images: [ "Melissa/Mowgli/mowgli_at_school.jpg",
            "Melissa/Mowgli/mowgli_in_moose_sweater.jpg",
            "Melissa/Mowgli/mowglis_first_steps.jpg",
            "Melissa/Mowgli/sleep_puppy.jpg" ]
}
```

Once the initial pet information is populated on the page, the #guess-btn should be enabled (it is initially disabled with the class "disabled" - removing this class will enable the button).

Processing Guesses

When a user clicks the guess button, the button should be disabled again and the currently-selected TA/instructor name in #ta-list should be used to make a guess of the pet's owner. If the guess is correct for the current pet, increment the value for #count by 1. Whenever a guess is made, #total should be incremented by 1.

Below is a screenshot after 6 guesses with 5 correct guesses (with a new random pet displayed):



CSE 154 Pets!

Can you guess which pet belongs to which instructor/TA?

Mowgli (Dog)

Correct guesses: **5**

Total guesses: **6**

After a guess has been made, a new random pet should be retrieved from pets.php to populate the #pet-info similar to case for the first random pet. The current selection in the drop-down should remain unchanged. Once data for a new random is retrieved successfully, re-enable the #guess-btn.

The HTML for the page is provided below. Start your JS solution on the next page.

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <script src="pets.js"></script>
    <link href="pets.css" rel="stylesheet" />
  </head>
  <body>
    <h1>CSE 154 Pets!</h1>
    <p>Can you guess which pet belongs to which instructor/TA?</p>

    <div id="pet-info">
      <h2><span id="pet-name"></span> (<span id="pet-type"></span>)</h2>
      <img class="hidden" id="pet-img" src="" alt="super cute pet" />
    </div>

    <div id="guess-ui">
      <p>Staff:</p>
      <select id="ta-list"></select>
      <button class="disabled" id="guess-btn">Guess!</button>
    </div>

    <div>
      <p>Correct guesses: <span id="correct">0</span></p>
      <p>Total guesses: <span id="total">0</span></p>
    </div>
  </body>
</html>
```

Write your solution to Problem 4 below:

```
(function() {
  "use strict";










})();
```

5. The Thing That We Fetch From For Fetching Pets Who Fetch (PHP)

Write a PHP web service called pets.php which provides data about CSE 154 staff and their pets. For this problem, assume your PHP file is in the same directory as pets.txt and a collection of folders for each TA/instructor. On the rest of this page, we will provide a short overview of the files/directories you will work with. The implementation requirements for the web service will be given on the following page.

pets.txt

This file contains information about each staff member's pet on its own line, in the following format:

```
name petname pettype petage
```

where name is the staff member's first name, petname is the name of the pet owned by name, pettype is the type of pet, and petage is the age of the pet. You may assume name, petname, and pettype contain no spaces and have only English alphabet letters, but petage may have spaces and numbers (e.g. "1 month" or "8 years"). Note that some staff members have more than one pet, but each pet is on its own line.

An example of pets.txt is given below:

```
Melissa Mowgli Dog 10 months
Kyle Pascal Dog 9 years
Sam Cachaca Dog 3 years
Lauren Spot Cat 16 years
Lauren Jack Cat 16 years
Lauren Whitney Cat 16 years
Jeremy Coloratura RainbowPony 154 days
Conner Bailey Dog 6 months
```

Staff and pet directories

Any staff member who has a pet has a directory named for each of their pets. Each pet directory contains at least one .jpg photo (it may also include non-jpg file types). For example, Melissa's dog Mowgli is her only pet. So in the folder Melissa/, there is a folder called Mowgli/ which contains the following files:

```
mowgli_at_school.jpg
mowgli_in_moose_sweater.jpg
mowglis_first_steps.jpg
mowglis_growth_chart.csv
sleepy_puppy.jpg
```

Manesh does not have a pet, so in the directory Manesh/, there are no directories for pets (although there may be other folders/files in his directory). When implementing your PHP, you may assume that the format of each staff member's name and their pet's name is exactly the format of the corresponding folder names.

Web Service Implementation

Your web service should accept two possible modes (case-insensitive):

mode=tas

If a mode of tas is passed as a GET parameter, your web service should output as plain text a list of all staff members who own pets, with each TA/instructor's name on its own line. Below is example output:

```
Conner
Jeremy
Kyle
Lauren
Melissa
Sam
```

mode=random

If a mode of random is passed as a GET parameter, your web service should output a JSON response with information about a random pet listed in pets.txt as is shown in Problem 2. The general format of the expected JSON response is provided below:

```
{
  name: "staffname",
  petname: "petname"
  type: "pettype",
  age: "petage",
  images: ["image01.jpg", "image01.jpg", ...]
}
```

Anything above in quotes should be replaced with the corresponding information unique to the randomly-chosen pet.

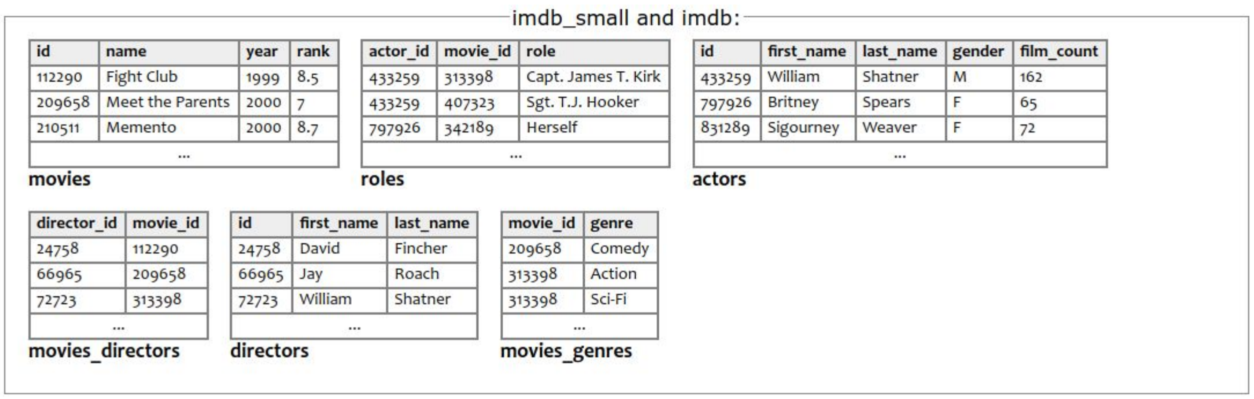
The array of images for the pet (the value for the JSON's images key) should contain all .jpg images found in the directory name/petname/ given as full paths relative to the location of pets.php. For example, if Mowgli was the random pet result for a call to `pets.php?mode=random`, the result JSON would be identical to that from Problem 3. Each pet in pets.txt should have an equally-likely chance of being returned from this mode.
If either mode is not passed, your program should output an error with a status of "`HTTP/1.1 400 Invalid Request`" and a plain text error message, "Error: Please pass in a mode parameter of tas or random."

Start your PHP code on the next page.

Write your solution to Problem 4 below:

6. SQL Queries

Recall the IMDB (Movies) database from the CSE 154 Query Tester:



imdb_small and imdb:

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

movies

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

roles

id	first_name	last_name	gender	film_count
433259	William	Shatner	M	162
797926	Britney	Spears	F	65
831289	Sigourney	Weaver	F	72
...				

actors

director_id	movie_id
24758	112290
66965	209658
72723	313398
...	

movies_directors

id	first_name	last_name
24758	David	Fincher
66965	Jay	Roach
72723	William	Shatner
...		

directors

movie_id	genre
209658	Comedy
313398	Action
313398	Sci-Fi
...	

movies_genres

1. Write a SQL query to list the first and last names of all the female actors that have a first name that starts or ends with “W”.

Expected results (5 rows returned total):

first_name	last_name
Wendy Lee	Avon
Wilma Jeanne	Cummins
Wynter	Kullman
W. Lauren	Sanchez
Meadow	Williams

2. Write a SQL query to list all columns out of the directors table for all directors that have directed a movie with a rank of 8 or higher, order by last name of director in alphabetical order. Return one row per unique director.

Expected results (20 rows returned total):

id	first_name	last_name
429	Andrew	Adamson
9247	Zach	Braff
11652	James (I)	Cameron
...	...	...