CSE 154 17sp Final Exam (Key)

Name: ________________________________ Quiz Section: __________
TA: ________________________________ Student ID #: ____________________

Rules: You have 110 minutes to complete this exam. You will receive a deduction if you keep working after the instructor calls for papers. This test is open-book and open note. 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. You may use JavaScript function aliases like `$` or `qs` **only** if you define them in your code. You **may** use the `AjaxGetPromise` and `AjaxPostPromise` objects in any JavaScript programs you write.

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!

Problem:	Possible:	Score:
HTML	10	
CSS	10	
JavaScript	16	
~~PHP~~	~~16~~	
~~JavaScript/Ajax~~ (POST requests not on 17au midterm)	~~16~~	
~~PHP/SQL~~	~~16~~	
~~Regular Expressions~~	~~6~~	
Short Answer	10	
Extra Credit	1	
	100	

*Note for midterm practice: Strike-through problems are not relevant to midterm exam content.

1. HTML Validation (10 points)

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <script type="text/javascript" src="index.js"></script>
        <link href="styles.css" rel="stylesheet"/>
    </head>
    <body>
        <h1 "Best page ever!" />
        <div>
            <a>Check out this cool page:
                <href>http://www.pointerpointer.com</href>
            </span>
        </div>
    <body>
</html>
```

This HTML document won't validate, and would generate errors and warnings in the W3C Validator. However, it is possible to make 5 modifications to the HTML to make it pass validation. Each modification might result in multiple text "changes" to the HTML document, but is considered one modification because it is addressing the same root problem.

Indicate the 5 modifications we need in order to make it pass validation. Write directly on the HTML. Below, briefly describe the changes that you made, and why you had to make each change in order to validate. If it is unclear what changes go with which explanations, feel free to number your changes on the HTML. No need for an essay — 10 words or less should be plenty.

```
1. Add a closing </a> tag

2. Text in <h1> should be within <h1> and </h1> opening/closing tags.

3. <href> is not a valid tag - should be attribute in <a>

4. Missing <title> in <head> section

5. Remove closing </span> or add matching opening <span>
```

2.a CSS Selectors (10 points total)

```
<body>
    <!-- Part 1: selection simulation -->
    <ul id="list-1">
        <li id="cookie-1">Shortbread</li>
        <li id="cookie-2">Thumbprint</li>
        <li id="list-id">
            <ol id="list-2">
                <li id="cookie-3">Oatmeal</li>
                <li id="cookie-4">Sugar</li>
            </ol>
        </li>
    </ul>
    <ol id="list-3">
        <li id="cookie-5">Pumpkin</li>
        <li id="cookie-6">Cherry</li>
    </ol>
</body>
```

Part 1. Write the ids of the elements selected by the given selectors:

1. `li`

cookie-1, cookie-2, list-id, cookie-3, cookie-4, cookie-5, cookie-6

2. `ol li`

cookie-3, cookie-4, cookie-5, cookie-6

3. `ul > li`

cookie-1, cookie-2, list-id

4. `ol, ul`

list-1, list-2, list-3

5. `#list-1 li`

cookie-1, cookie-2, list-id

2.b CSS Selectors (10 points total)

```
<body>
    <!-- Part 2: write the selector -->
    <main>
        <article> (A)
            <article> (B)
                <section></section> (C)
            </article>
            <section class="clinton"></section> (D)
        </article>
        <section class="trump"></section> (E)
        <section> (F)
            <article class="trump"> (G)
                <section></section> (H)
                <article></article> (I)
                <section> (J)
                    <section></section> (K)
                </section>
            </article>
        </section>
        <aside></aside> (L)
    </main>
</body>
```

Part 2: Write CSS Selectors that select the following elements. Your selector must not select other elements in the document. The tags are lettered on the line that they are opened.

```
1. D
.clinton

2. E, G
.trump

3. A, B, C, D, E, F, G, H, I, J, K
article, section

4. H, J, K
section section

5. K, L
section > section, aside
```

3. JavaScript (16 points)

Use the following HTML and CSS as reference for Problem 3.

```
HTML:
…
<head>
    …
    <script type="text/javascript" src="graph.js"></script>
    <link rel="stylesheet" href="graph.css"/>
</head>
<body>
    <div id="graph"></div>
    <button>New point!</button>
</body>
…

graph.css:
#graph {
    position: relative;
    width: 600px;
    height: 600px;
    margin-left: auto;
    margin-right: auto;
    border: 2px solid black;
}

.point {
    position: absolute;
    width: 8px;
    height: 8px;
    border-radius: 5px;
    border: 1px solid black;
}
```

3. JavaScript (16 points)

Given the provided HTML and CSS, implement `graph.js`. After the window loads, when a user clicks the 'New Point!' button, you are to add a new point `div` into the `#graph` element at a random position within the #graph.

Additionally, when you create a point, you are to set the color of the point based on the top and left coordinates that you randomly picked for the location. The green color component of all the points is 0. The red component of the color is determined by the horizontal position, and the blue component is determined by the vertical position of the point. The red and blue components start at a value of 0 (in points at the left and bottom edges, respectively), and have a max value of 255 (in points at the right and top edges, respectively), and scale linearly in between.

Requirements:

- each point is to appear at a random location in the `#graph`
- both the top and left coordinates must be integers
- the circle representing each point must fit entirely within the `#graph div`
- you may not change the `#graph div` in any way, besides appending point elements inside it
- every possible (integer pair) location must have the same chance of a point appearing there
- the color of a point is based off of the coordinates of it's top/left location
- when any point is double-clicked, it should be removed from the page
- you must correctly encapsulate your JavaScript in a module

Hints:

- this problem is harder if you try to calculate/work with the center-points of the circles. It is much easier if you only think about a point as its top/left coordinates
- though, remember that the points in the upper right corner of the `#graph` are the most blue and the most red
- the CSS `rgb()` color value doesn't work with floating point numbers -- it needs integers
- to figure out if a circle fits inside the `#graph`, determine if a square that circumscribes the circle fits in the `#graph`

(space for problem 3)

Solution:
```
“use strict”;
(function() {
  window.onload = function() {
    let button = document.querySelector("button");
    button.onclick = newPoint;
  };

  function newPoint() {
    let point = document.createElement("div");
    point.className = "point";
    let x = Math.random();
    let y = Math.random();

    // subtract 8 to account for diameter of point
    point.style.left = x * (600 - 8) + "px"; // don't forget the units!
    point.style.top = y * (600 - 8) + "px";
    let red = Math.round(x * 255);
    // inverse blue ratio for distance from
    let blue = Math.round(255 - (y * 255));

    point.style.backgroundColor = "rgb(" + red + ", 0, " + blue + ")";
    document.getElementById("graph").appendChild(point);
  }
})();
```

~~4. PHP Web Service (16 points)~~ **Writing PHP Not on Midterm**

~~Implement a PHP web service `pw-check.php` that validates a password, and protects itself by refusing requests if too many attempts are made in quick succession.~~

~~Accept a POST parameter with the key `password`: if the given password is correct, then your PHP code should produce a status code 200 with a plain text response of "`success`". (There are much better ways to store and validate passwords, but for this question, you can simply check if the given password is the following string: "`123456`"). If the given password is incorrect, your code should produce a status code 200 with a plain text response of "`failed`".~~

~~Your web service is to implement a crude form of protection against someone guessing the password. Specifically, when more than 10 failed password attempts have been made in the last 10 seconds, then your web service should enter a state where it rejects **the current request and all future** requests with a status code of 429 (Too Many Requests) and produce no output.~~

~~A 'failed password attempt' means a request in which the client passed `password`, but it was incorrect, and the response that was generated was "`failed`".~~

~~Note that we don't have a good way of determining where any of the requests are coming from. This means that we can't distinguish between 1 client making 11 failed attempts, and 11 different clients making 1 failed attempt each. So no matter where the attempts come from, if we have more than 10 attempts in 10 seconds, then the server is to refuse all requests.~~

~~The allowed rate of 10 failed attempts every 10 seconds is allowed to be exceeded in short bursts, for example, 7 failed attempts in 1 second should not cause the server to refuse requests. Only once more than 10 failed attempts have been made should you consider refusing requests.~~

~~If the client does not pass the `password` parameter, then you should respond with a status code 400 with a plain text message of "`password parameter required`".~~

~~Hints:~~

- ~~Use the `time()` function in PHP to get the current time (in seconds)~~
- ~~Use files to persist information about failed attempts and/or the state of the webserver between requests~~

~~5. JavaScript/Ajax (16 points)~~ **POST Requests Not on Midterm**

~~Write a JavaScript program `guess-pw.js` that calls out to `pw-check.php` and attempts to guess the password. You know that the password is 6 numerical digits long (112358, 072229, 000000, etc), but you don't know what the password is. You also know that the server will lock you out if you call it too often, so you'll need to prevent your JavaScript from calling the web service too frequently.~~

~~Your JavaScript client is to periodically `POST` to the `check-pw.php` web service with a parameter named `password` until the the server responds with the text "`success`".~~

~~POST one password guess at a time. Test passwords systematically.~~

~~Call the server in such a way to keep it from locking up, meaning you may not make more than 10 requests in 10 seconds. Call the server fast enough to find the password in a reasonable time. An average rate of 10 requests / minute can try all of the combinations within 10 weeks. Your JavaScript code must determine the password in 10 weeks time or less.~~

~~*Details:*~~
~~If the server responds with a 200 and text "`success`", this means that you have guessed the password. Log the following message to the console: "`found it: <password>`" (replacing `<password>` with the correct password). Cease making requests to `pw-check.php`.~~

~~If the server responds with a 200 and text "`failed`", this means that you have guessed an incorrect password. Log the following message to the console: "`incorrect: <password>`" (replacing `<password>` with the incorrect guess).~~

~~If the server returns status code 200, you may assume that it also produced a message of either "`success`" or "`failed`".~~

~~If the server responds with a status code of anything besides 200, this means an error has occurred: your client is to cease making any requests to `pw-check.php`.~~

~~You may assume for this question that your JavaScript file and the PHP web service are stored on the same server in the same directory.~~

~~You may assume that the `AjaxGetPromise` and `AjaxPostPromise` objects are available.~~

~~Note that if the password is "000123", and you pass in a value of "123", the password check will return "`failed`".~~

~~Assume a correct and functional implementation of the PHP web service. Assume that the PHP web service can process requests and return responses much faster than you are making requests.~~

(space for problem 5)

~~6. PHP/SQL (16 points)~~ PHP and SQL Not on Midterm

~~Write a PHP program called `snapshot.php` that pulls some data out of a database, aggregates it to find the average of one of the values, and inserts some summary values (the count, and the average) into another table.~~

~~There is a table called `temperatures` that contains daily temperature highs:~~

~~date~~	~~hightemp~~
~~2017-01-01~~	~~34~~
~~2017-01-02~~	~~33~~
~~...~~	~~....~~
~~2017-05-30~~	~~75~~

~~There is another table called `snapshots` that contains the average of all of the highs in `temperatures`, taken at various points in time.~~

~~count~~	~~average~~
~~23~~	~~32.1~~
~~24~~	~~32.9~~
~~...~~	~~...~~
~~290~~	~~48.123~~

~~Each row in `snapshots` contains the average at the moment that the snapshot was taken, and the count of rows in `temperatures` when the snapshot was taken.~~

~~Implement `snapshot.php` in which you are to `SELECT` all of the rows out of `temperatures`, take the mean of the `hightemp` column across all of the rows, and then `INSERT` a row into `snapshots` with the count of rows found in `temperatures` and the average `hightemp` of the rows.~~

~~Every time your `snapshot.php` program is executed, it should insert exactly 1 new row into `snapshots`. If you were to call `snapshot.php` twice without updating `temperatures`, you would insert a duplicate row into `snapshots`. This is expected.~~

~~Your PHP program connect to the SQL database using a PDO object. The MySQL instance holding your database is located on the same server as your PHP program. Connect to the database named "`weather`" with username "`stevepoole`" and password "`mete0RZ`".~~

~~7. Regular Expressions (6 points)~~ Reg. Expressions Not on Midterm

~~Write regular expressions that match the following descriptions. Note that the `‘quotation’` `‘marks’` in the example are to show you where strings start and end — they are not part of the string. **Bolded** text indicates which portion of the matching strings matched.~~

~~1. Binary String: a non-empty sequence of ‘0’ or ‘1’ characters. Match only if the whole input string is a binary string.~~

~~String that match:~~	~~No match:~~
~~‘**0011**’~~	~~‘onezeroone’~~
~~‘**1010101**’~~	~~‘’~~
~~‘**00011100011110**’~~	~~‘1001 ’~~

~~1.~~ ______________________________

~~2. UW NetID: at least 1 and at most 8 lowercase letters or numbers. Must start with a lowercase letter. Match only if the whole input string is a UW NetID.~~

~~Strings that match:~~	~~No match:~~
~~‘**whitab1**’~~	~~‘88keys’~~
~~‘**medskm23**’~~	~~‘aabbccddf’~~
~~‘**z0rr0**’~~	~~‘super!’~~

~~2.~~ ______________________________

~~3. INFO/CSE Courses: the uppercase department abbreviation (either INFO or CSE), followed by three digits, with no spaces or other characters in between. Match if a course number is found anywhere in the input string:~~

~~Strings that match:~~	~~No match:~~
~~‘**INFO999**’~~	~~‘INF100’~~
~~‘I am taking **CSE142** this quarter’~~	~~‘ICNSFEO200’~~
~~‘99**INFO340** is a great follow up to CSE__!!154’~~	~~‘899CSE’~~

~~3.~~ ______________________________

8. Short answer questions (10 points, 1 each)

a. When making web service calls with JavaScript, why is it preferable to use asynchronous calls (as opposed to synchronous)?

   Solution: Execution of a program can continue without waiting for an asynchronous web service call to finish.

~~b. What are Promises for? For example, we implemented some Promises in class to work with Ajax calls. What about Ajax calls makes them good candidates to be run inside Promises?~~

~~c. If your web application accesses a database, is the code that (directly) accesses the database going to be running on the client (in the browser) or on the server?~~

~~d. Why do we validate user input on the server side?~~

e. Why do we validate user input on the client side?
   To provide immediate feedback to users when they provide invalid input, and to reduce the chance of sending malicious input to the server.

f. Is it appropriate to pass query parameters the URL (as in: `foo.php?k=v&k2=v2`) during a GET or POST request? (pick either GET or POST)
   GET

g. Why do we use the module pattern when writing JavaScript code?
   To improve modularity between different JavaScript files and to avoid cluttering the global namespace.

h. When is it appropriate to use a class (as opposed to an ID) in HTML?
   When multiple elements share common styles.

~~i. Where is "cookie" data stored?~~

~~j. Where is "session" data stored?~~

9. Extra Credit (1 extra credit point)

If your TA were an HTML tag, which one would they be, and why? You may also give us an artistic rendering of your TA as an HTML tag, if you prefer. (Drawing, poetry, etc). Any work that appears to have taken more than 1 minute of effort and is not offensive/inappropriate will receive the extra credit point.