CSE 154, Spring 2014
Final Exam, Tuesday, June 10, 2014

Name: ____

Quiz Section: ____________________ **TA:** ____________________

Student ID #: ____________________

Rules:

- You have **110 minutes** to complete this exam.
  You may receive a deduction if you keep working after the instructor calls for papers.
- This test is open-book, but closed notes. You may *not* use printed/written notes or 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 dot-dot-dot marks (...).
  You may write **`ID`** for `document.getElementById` and **`QS`** for `document.querySelectorAll`.
- 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.

Good luck! You can do it!

('O')

Problem	**Description**	**Earned**	**Max**
1	HTML / CSS Tracing		20
2	PHP		20
3	JS		20
4	JS / Ajax / JSON		20
5	SQL		20
X	Extra Credit		1
TOTAL	**Total Points**		**100**

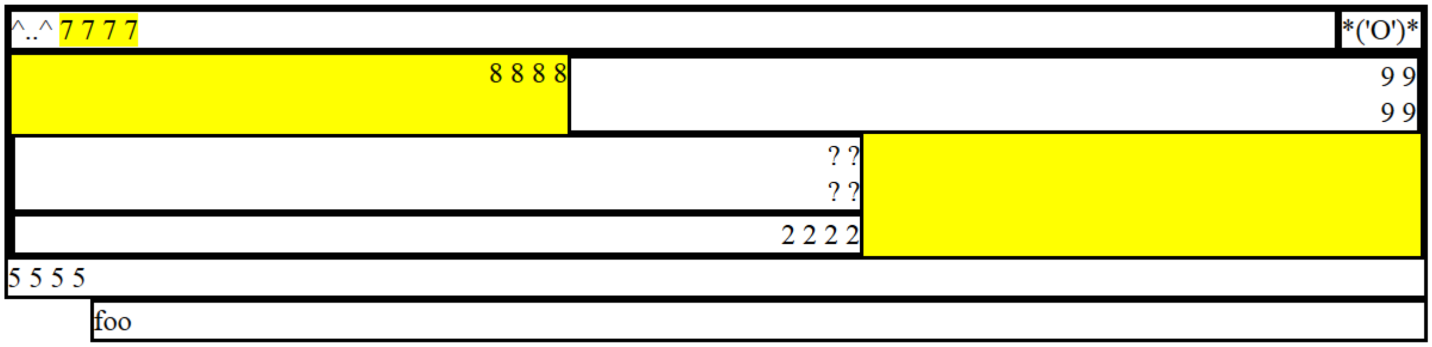
1. HTML / CSS Tracing

Draw a picture of how the following HTML/CSS code will look when the browser renders it on-screen. Assume that the HTML is wrapped in a valid full page with a `head` and `body`. Indicate a non-white background by shading lightly or by drawing diagonal lines like this. It is possible that some CSS rules shown will not apply to any elements.

```html
<div>
  <div class="id">*('O')*</div>
  <div>
    <span id="id">^..^</span>
    <span class="class">7 7 7 7</span>
  </div>

  <div class="class">
    <span>8 8 8 8</span>
    <div class="id">9 9<br />9 9</div>
    <div id="class">? ?<br />? ?</div>
    <div>2 2 2 2</div>
  </div>
  <span id="five">5 5
                5 5</span>
</div>
<div id="foo">foo</div>
```

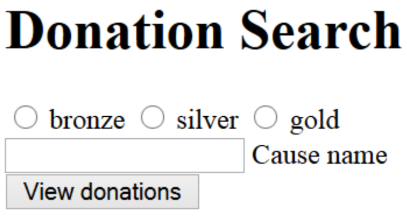
```css
div { border: black 2px solid;
        overflow: hidden;}
.class {
      text-align: right;
      background-color: yellow;}
.class > div {width: 60%;}
div div > div {background-color: white;}
.id { float: right;}
#id {float: left;}
#five{text-align: right;}
#foo{margin-left: 3em;}
```



2. PHP

Write the PHP code for a web page `donations.php` that displays donors who have donated amounts in a specified range. Assume that there is a provided page with a form where the user can type the name of a cause and select a donation level. The form will submit to your `donations.php`. Here is the relevant HTML from that provided page, and a screenshot of its appearance:

```
<h1>Donation Search</h1>
<form action="donations.php" method="get">
  <div>
    <input type="radio" name="level" value="bronze"/> bronze
    <input type="radio" name="level" value="silver"/> silver
    <input type="radio" name="level" value="gold"/> gold
  </div>
  <div><input type="text" name="cause" /> Cause name</div>
  <div><input type="submit" value="View donations" /></div>
</form>
```



Each cause has a corresponding text file in the current directory that contains all donations for that cause. The file may not have the exact name of the cause but it will contain the exact name. For example, if the user types "kermit" it should match "presidentkermitfrog.txt". If there is more than one file that contains the name of the cause you should select the first one. Each line in the file contains the donator's name, the donation amount and "`yes`" if the donation was matched, "`no`" otherwise. Each item is separated from the others by a semicolon (":"), as in the example below at right.

The bronze donation range is amounts between 0 and 100 inclusive, silver between 100 exclusive and 500 inclusive and gold above 500.

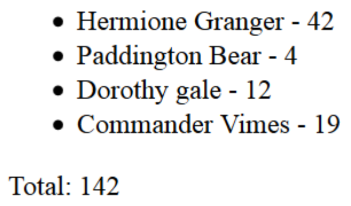
```
Hermione Granger:42:yes
Costco:5000:no
CSE 154 Students:126:no
Trillian:60000:yes
Paddington Bear:4:yes
```

Your PHP page should accept the query parameters from the above form, open the text file for the correct cause, and search it for donations in the range matching the selected category.

Display each donation as a bullet in an unordered list, showing the donor and the amount separated by a dash (-). At the bottom of the page output the total amount donated from this group. If the donation was matched, the amount donated should be added to the total twice.

The following screenshots show the PHP page output after the user submits the form with various values:

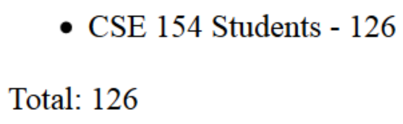
1. Bronze – cause: Kermit for president



- Hermione Granger - 42
- Paddington Bear - 4
- Dorothy gale - 12
- Commander Vimes - 19

Total: 142

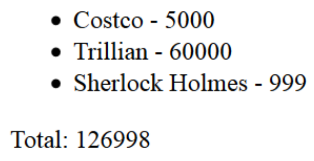
2. Silver level - cause: Kermit for president



- CSE 154 Students - 126

Total: 126

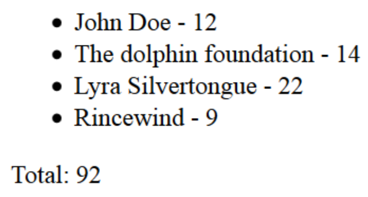
3. Gold level – cause: Kermit for president



- Costco - 5000
- Trillian - 60000
- Sherlock Holmes - 999

Total: 126998

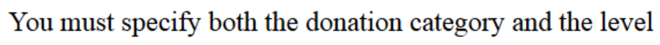
4.Bronze – cause: Pooh for president



- John Doe - 12
- The dolphin foundation - 14
- Lyra Silvertongue - 22
- Rincewind - 9

Total: 92

5. No level selected



You must specify both the donation category and the level

You may assume that the user will submit parameters with the appropriate values (e.g. the level will always be gold, silver or bronze) if they submit parameters. However, they may omit parameters. If they do your code should print an error message explaining what went wrong. You may assume that a file for the cause the user submits exists and is valid in the format described above. You can write just the code that would go inside the page `body`; you don't need to output a `head` section or a complete page. Use the browser's default styling; do not write any CSS for this problem.

Write your answer on the next page.

2. PHP (writing space)

```
<?php
if (isset($_GET["level"]) && isset($_GET["cause"])) {
        $level = $_GET["level"];
        $cause = $_GET["cause"];

        $files = glob("*$cause*");
        $file = file($files[0]);//, FILE_IGNORE_NEW_LINES);
        $total = 0;
        ?>
        <ul>
        <?php
        foreach($file as $line) {
                list($name, $amount, $match) = explode(":", $line);
                        if (($level == "bronze" && $amount >= 0 && $amount <= 100) ||
                            ($level == "silver" && $amount > 100 && $amount <= 500) ||
                            ($level == "gold" && $amount > 500)) {

                                $total += $amount;
                                if ($match == "yes") {
                                        $total += $amount;
                                } ?>
                                <li><?= $name ?> - <?= $amount ?></li>
                                <?php
                        }
        }
        ?>
        </ul>
        <p>Total: <?= $total ?></p>
<?php
} else {
?>
        <p>You must specify both the donation category and the level</p>
<?php
}
?>
```

3. JavaScript / DOM

Write the **JavaScript code** to add behavior to the following page for keeping track of a to-do-list. The page UI allows the user to type an item into a text box. The user can click the "add" button to add the item to the bottom of the list. Each word in the phrase should be inserted as a `li`, inside a `ul` with the `id` of `list`.

If the user wishes to remove an item he or she can type the text of the item he or she wishes to remove in the text box and click the "remove" button. This should be case insensitive. For example, if the list only contains "foo" and the user tries to remove "FoO", it should be removed. If the user tries to remove an item that is in the list multiple times only the first occurrence should be removed.

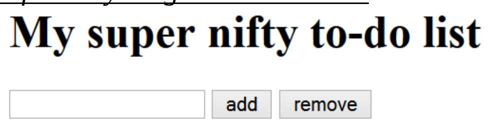
The items should have background colors that alternate between white and yellow (first white, then yellow, then white, yellow, etc.). This should still be the case no matter how many items are removed or added and no matter what order these operations are done in. You **may not** use the CSS3 nth-child pseudo selector to do this.

The code should work for multiple clicks of the buttons. On each click it should clear any previous information you typed in the input boxes. **Do not use any JavaScript libraries such as jQuery or Prototype**. Here is the relevant HTML code for the page:

```
<h1>My super nifty to-do list</h1>
<ul id="list"></ul>
<div>
   <input type="text" id="item" />
  <button id="add">add</button>
  <button id="remove">remove</button>
</div>
```

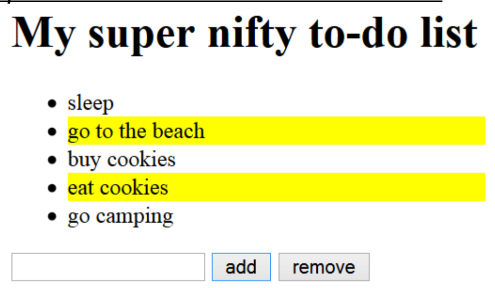
These screenshots show the state after items have been added, and the state after items have been removed.

1. *Before anything has been added*



My super nifty to-do list

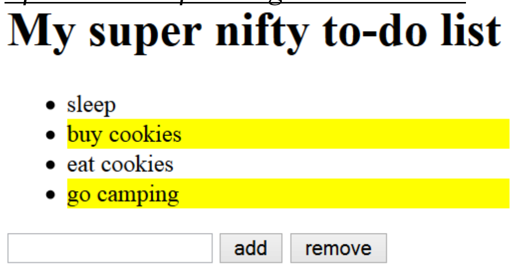
2. *After 5 items added and none removed*



My super nifty to-do list

- sleep
- go to the beach
- buy cookies
- eat cookies
- go camping

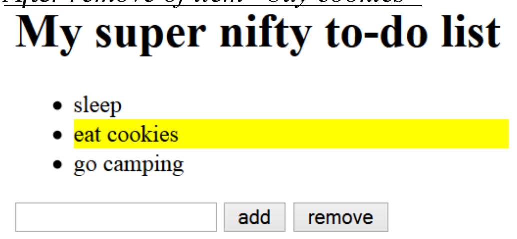
3. *After remove of item "go to the beach"*



My super nifty to-do list

- sleep
- buy cookies
- eat cookies
- go camping

4. *After remove of item "buy cookies"*



My super nifty to-do list

- sleep
- eat cookies
- go camping

Write your answer on the next page.

3. JavaScript / DOM (writing space)

```
var yellow = false;

window.onload = function() {
        document.getElementById("add").onclick = addItem;
        document.getElementById("remove").onclick = removeItem;
}

function addItem() {
        var item = document.createElement("li");
        item.innerHTML = document.getElementById("item").value;
        document.getElementById("list").appendChild(item);
        document.getElementById("item").value = "";
        if(yellow) {
                item.style.backgroundColor = "yellow";
        }
        yellow = !yellow;
}

function removeItem() {
        var item = document.getElementById("item").value;
        var list = document.querySelectorAll("li");
        document.getElementById("item").value = "";
        var found = false;
        for(var i = 0; i < list.length; i++) {
                if(list[i].innerHTML.toLowerCase() == item.toLowerCase() && !found) {
                        document.getElementById("list").removeChild(list[i]);
                        found = true;
                }
                if(found) {
                        if(i % 2 == 0) {
                                list[i].style.backgroundColor = "yellow";
                        } else {
                                list[i].style.backgroundColor = "white";
                        }
                        yellow = !yellow;
                }
        }
}
```

4. Ajax/JSON

Suppose that there is a web service named `flights.php`, located on your web server in the same directory as your code. This service outputs JSON data describing flights between various cities. In this problem you will write Ajax JavaScript code to contact the web service (using a GET request), examine its JSON data, and display a list of possible flight prices and carriers.

The page contains two textboxes where the user can specify the start location and end location, a check box that they can check if they want to only see non-stop flights (flights with 0 stops) and a “Go!” button. When the button is pressed you should send an Ajax request passing the parameter of “start”. You can assume that the user has typed a valid location into each box before pressing the button

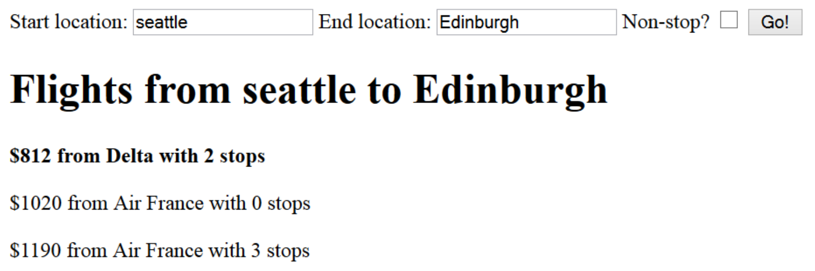
The JSON data returned by the web service consists of a list of end locations, each of which has a list of flights associated with it. The list of flights contains lists which each contain a price, a carrier and the number of stops.

```
{
   "Edinburgh":{
         "start":"Seattle",
         "flights":[{"carrier":"Delta","price":812,"stops":2},
                    {"carrier":"Air France","price":1020,"stops":0},
                    {"carrier":"Air France","price":1190,"stops":3}]
   },
   "New York":{
         "start":"Seattle",
         "flights":[{"carrier":"British Airlines","price":782,"stops":1},
                    {"carrier":"Delta","price":1562,"stops":2},
                    {"carrier":"United","price":957,"stops":1},
                    {"carrier":"KLM","price":687,"stops":3},
                    {"carrier":"KLM","price":1458,"stops":1}]
   }
}
```

The relevant existing HTML in the page is the following:

```
<div>
  <label>Start location: <input type="text" id="start" /></label>
  <label>End location: <input type="text" id="dest" /></label>
  <label>Non-stop? <input type="checkbox" id="stops"></label>
  <button id="go">Go!</button>
</div>
<div id="results"></div>
```

When the “Go!” button is clicked, clear previous results and read the JSON data with Ajax. Add a `h1` to the `results` div containing the text “Flights from” and then the start and destination locations. Turn each flight's data into a paragraph in the `results` div. In each paragraph, write the price of the ticket (with a “$”) followed by the word “from”, the carrier’s name and then the word “with”, the number of stops and the word “stops”. If the flight has a price below 1000 display its row in bold. For the example JSON shown above, the page is shown twice below, once with only non-stop flights and once with all flights.

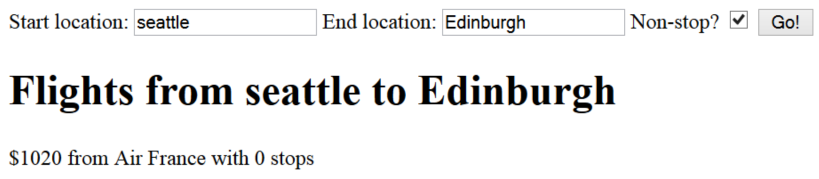

Start location: seattle End location: Edinburgh Non-stop? ☐ Go!

Flights from seattle to Edinburgh

$812 from Delta with 2 stops

$1020 from Air France with 0 stops

$1190 from Air France with 3 stops


Start location: seattle End location: Edinburgh Non-stop? ☑ Go!

Flights from seattle to Edinburgh

$1020 from Air France with 0 stops

You may assume that the JSON data is valid in the format described previously, the data typed into the text boxes is valid, and that the .php service is reachable. **You may not use any Javascript libraries such as Prototype and JQuery.**

Write your answer on the next page.

4. Ajax/JSON (writing space)

```
window.onload = function() {
        document.getElementById("go").onclick = req;
}

function req() {
        var start = document.getElementById("start").value;
        var ajax = new XMLHttpRequest();
        ajax.onload = display;
        ajax.open("GET", "flights2.php?start=" + start, true);
        ajax.send();
}

function display() {
        var noStops = document.getElementById("stops").checked;
        var start = document.getElementById("start").value;
        var dest = document.getElementById("dest").value;
        var div = document.getElementById("results");
        div.innerHTML = "";
        var ajax = JSON.parse(this.responseText);
        var title = document.createElement("h1");
        title.textContent = "Flights from " + start + " to " + dest;
        div.appendChild(title);

        var myAjax = ajax[dest];
        for(var i = 0; i < myAjax["flights"].length; i++) {
                var flight = myAjax["flights"][i];
                if(flight["stops"] == 0 || !noStops) {
                        var p = document.createElement("p");
                        p.textContent = "$" + flight["price"] + " from " +
                                        flight["carrier"] + " with " + flight["stops"] +
                                        " stops";
                        if(flight["price"] < 1000) {
                                p.style.fontWeight = "bold";
                        }
                        div.appendChild(p);
                }
        }
}
```

5. SQL

Write an SQL query to search the `world` database for all languages that are spoken as the official language of at least two "newly growing" countries. We will define a "newly growing" country as a country that has both of the following qualities: became independent after the year 1900, and contains at least one city with a population of over one million. For example, Malay would be listed because it is the official language of Malaysia which contains Kuala Lumpur (population 1,297,526) and became independent in 1957, and Indonesia which contains Jakarta (population 9,604,900) and became independent in 1945. Each language should be listed alphabetically and only once**.**

Recall the `world` tables:

code	name	continent	independence_year	population	gnp	head_of_state	...
AFG	Afghanistan	Asia	1919	22720000	5976.0	Mohammad Omar	...
NLD	Netherlands	Europe	1581	15864000	371362.0	Beatrix	...
...							

countries

country_code	language	official	percentage
AFG	Pashto	T	52.4
NLD	Dutch	T	95.6
...			

languages

id	name	country_code	district	population
3793	New York	USA	New York	8008278
1	Los Angeles	USA	California	3694820
...				

cities

```
+----------+
| name     |
+----------+
| Arabic   |
| Chinese  |
| English  |
| French   |
| German   |
| Korean   |
| Malay    |
| Russian  |
| Spanish  |
+----------+
9 rows in set (69 ms)
```

When run on `world` database, your query produces the results at left.

If you join too many tables together that are not needed for the query, you will not receive full credit. You should solve this problem using only the SQL syntax shown in class and the textbook.

```
SELECT DISTINCT l.language FROM languages l
JOIN languages l2 ON l.language = l2.language
JOIN countries co1 ON l.country_code = co1.code
JOIN countries co2 ON l2.country_code = co2.code
JOIN cities ci1 ON co1.code = ci1.country_code
JOIN cities ci2 ON co2.code = ci2.country_code
WHERE ci1.population > 1000000 AND ci2.population > 1000000
  AND co1.code <> co2.code AND l.official = TRUE AND l2.official = TRUE
  AND co1.independence_year > 1900 AND co2.independence_year > 1900
ORDER BY l.language
```

X. Extra Credit

Draw a picture of your TA as a superhero.

(This is just for fun; any picture that appears to reflect more than a few moments' work will receive credit.)