CSE 154, Autumn 2014
Final Exam, Tuesday, December 9, 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		20
2	Javascript		20
3	PHP		22
4	Ajax		20
5	Regular Expressions		18
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  .

a)

```
<div class="baz">
    <div id="bar">
        <div>$$</div>
        <p id="foo">text</p>
         other stuff
    </div>
</div>
```

```
div > div { float: right; }
div {
   border: black solid 1px;
   padding: 5px;
}
.baz {
   background-color: yellow;
   overflow: hidden;
}
```

b)

```
<div>
    <div class="cat">^..^</div>
    <div id="dog">dog</div>
    <div id="mouse">O..O</div>
</div>
```

```
div { border: 1px black solid; }
div { text-align: right; }
.cat {
   text-align: center;
   padding: 20px;
}
#dog { text-align: left; }
#mouse { margin: 40px; }
```

2. Javascript/DOM

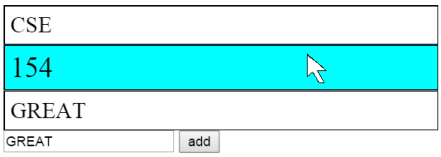
Write JavaScript code in a file called `simple.js` for manipulating a list. The page UI allows the user to type some text into a text box. The user can click an "`add`" button to make the text appear in the list (put it inside the `game` div).

When the mouse enters a list item the background color of that item should turn `cyan` and the `font size` of that list item should increase by 5px. When it leaves that item the background color should turn back to `white` and the `font size` back to what it originally was. **The original font size is included in a CSS file attached to the HTML**. Your code should work for any starting font size value. You can view the current size with your code but not by opening the file and looking at it.

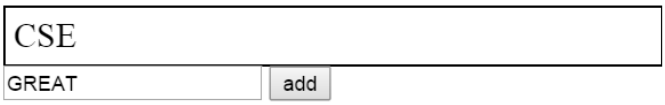
When the user clicks on an item it should be **deleted**.

You may not use any Javascript libraries such as Prototype and JQuery. You must write ALL Javascript to make the page work, none is provided. You can define any additional CSS you like as long as you label it in a clear way. You only need to write the appearance/CSS changes mentioned above. You can assume the page has a linked style sheet that provides the other styles visible in the screenshots.

The page when it first loads



The page after a couple adds with the mouse hovering



The page after "154" and "Great" have been clicked and removed

Your Javascript will be manipulating the following HTML code:

```
<!DOCTYPE html>
<html>
	<head>
		<script type="text/javascript" src="simple.js"></script>
		<link rel="stylesheet" type="text/css" href="simple.css" />
	</head>
	<body>
		<div id="game"></div>
		<input id="input" type="text" />
		<button id="add">add</button>
	</body>
</html>
```

(Write your solution on the next page)

2. Javascript /DOM (Additional writing space)

3. PHP/JSON

Write the code for a web service `courses.php` that outputs `JSON` data about available courses at a school. This service should take two `GET` parameters named `start` and `end`, and search a data file for all courses that match those start/end times exactly and have open seats available.

Your `PHP` code will read a data file named `courses.txt`. Each line in that file matches the following format, with each token of information separated by a single space:

code startTime endTime seatsAvailable seatsTotal name

All tokens of data except the course name are guaranteed not to contain any spaces in them. You can assume all data in the file is valid, a number when you expect it to be a number and has no blank or malformed lines.

You may find an optional third parameter to the `split` function useful when writing your solution. If you pass `split` a number as a third parameter it will cap the number of times it splits to that number. Everything after the number of splits is passed will be placed in the last spot in the array. For example `split(":", "h:i:j:k", 3)` would return `{"h", "i", "j:k"}`.

The `JSON` that is output should contain a number labeled "`count`". This should represent a count of all courses at the given start and end times. It should also contain a list called "`courses`" that contains a list for each course exactly matching the start and end times that has open seats. The list for each course should contain the code labeled as "`code`", the number of seats left (the total seats minus the seats available) labeled as "`seats`", and the name labeled as "`name`",

You can assume both parameters will be passed. If no courses match the start/end and have seats you should send back the same data as usual, just leave the course array empty.

An example input file:

```
CSE154 130 230 250 250 Web Programming
CSE143 130 230 700 800 Computer Programming I
ANTH300 130 230 13 14 Anthropology
DANCE250 130 3 40 50 World Dance History
```

Output using the above input file and `courses.php?start=130&end=230`

```
{ "count":3,
  "courses":[ {"code" : "CSE143",
                 "seats" : 100,
                 "name" : "Computer Programming"
               },
               {"code" : "ANTH300",
                 "seats" : 1,
                 "name" : "Anthropology"
               }
             ]
}
```

3. PHP /JSON (additional writing space)

4. Ajax/JSON

Write Javascript code in a file called `courses.js` that contacts the `courses.php` code that you wrote for question 3. You can assume `courses.php` and `courses.js` will be located in the same directory. Remember, `courses.php` takes `start` and `end` times as parameters and outputs data in the form of the data below:

```
{ "count":3,
  "courses":[ {"code" : "CSE143",
               "seats" : 100,
               "name" : "Computer Programming"
              },
              {"code" : "ANTH300",
               "seats" : 1,
               "name" : "Anthropology"
              }
            ]
}
```

Your Javascript code will be attached to the below `HTML` page:

```
<!DOCTYPE html>
<html>
   <head><script type="text/javascript" src="courses.js"></script></head>
   <body>
         <h1>Search for open courses</h1>
         <label>start time:<input id="start" type="text" /></label>
         <label>end time:<input id="end" type="text" /></label>
         <button id="search">search</button>
         <ul id="results"></ul>
         <p id="count"></p>
   </body>
</html>
```

When the `search` button is clicked, clear the previous results on the page and request the `JSON` data for the input times with `Ajax`. Add each returned course to the list in the following format:

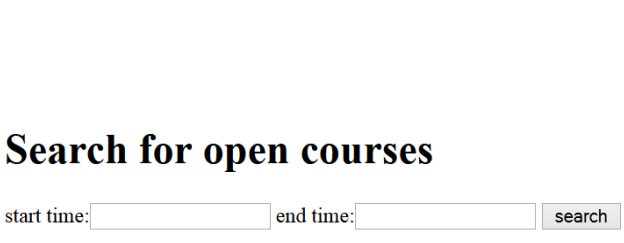
name - ***seats*** *seats*

Add the count to the `count` paragraph in the following format:

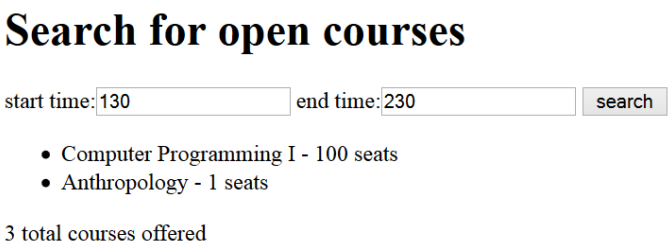
count *total courses offered*

You may assume that the JSON data is valid and in the format described previously, the data typed into the text boxes is valid, and that the `.php` service is reachable. You do not need to do anything special if there are no matching courses.

You may not use any Javascript libraries such as Prototype and JQuery.



When the page first opens



After a search for courses 130-230

(Write your solution on the next page)

4. Ajax/JSON (additional writing space)

5. Regular Expressions

a) Write a regular expression to match a **hexadecimal color code**. Remember, colors written in hexadecimal always start with a `#` and then contain 6 letters or numbers `0-9` and `A-F`. Letters can be lowercase or uppercase.

match:	**don't match:**
`#000000`	`#1234567`
`#D3D3D3`	`4#D3D3D3`
`#abCDeF`	`#abCDeG`

b) Write a regular expression to validate a **Mastercard number**. Mastercards have a 16 digit long number. The first number is always `5` and the second number is a `1`, `2`, `3`, `4` or `5`. The rest of the numbers can be anything. You should not look for or try to match dashes (-).

match:	**don't match:**
`5112345678901234`	`51123456789012`
`5555555555555555`	`55555a5555555b55`

c) Write a regular expression to match a **time** written like `11:04 AM`. Times consist of an hour `(1-12)` followed by a colon, followed by minutes `(00-59)`, followed by a space and then either `AM` or `PM`.

match:	**don't match:**
`12:00 AM`	`12:0 AM`
`1:11 PM`	`14:00 PM`
`4:59 PM`	`0:20 AM`
	`02:00 AM`
	`4:60 PM`

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

X. Extra credit

Draw a picture of your TA as a superhero.

(any answer that reflects non-trivial effort will receive credit)