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.

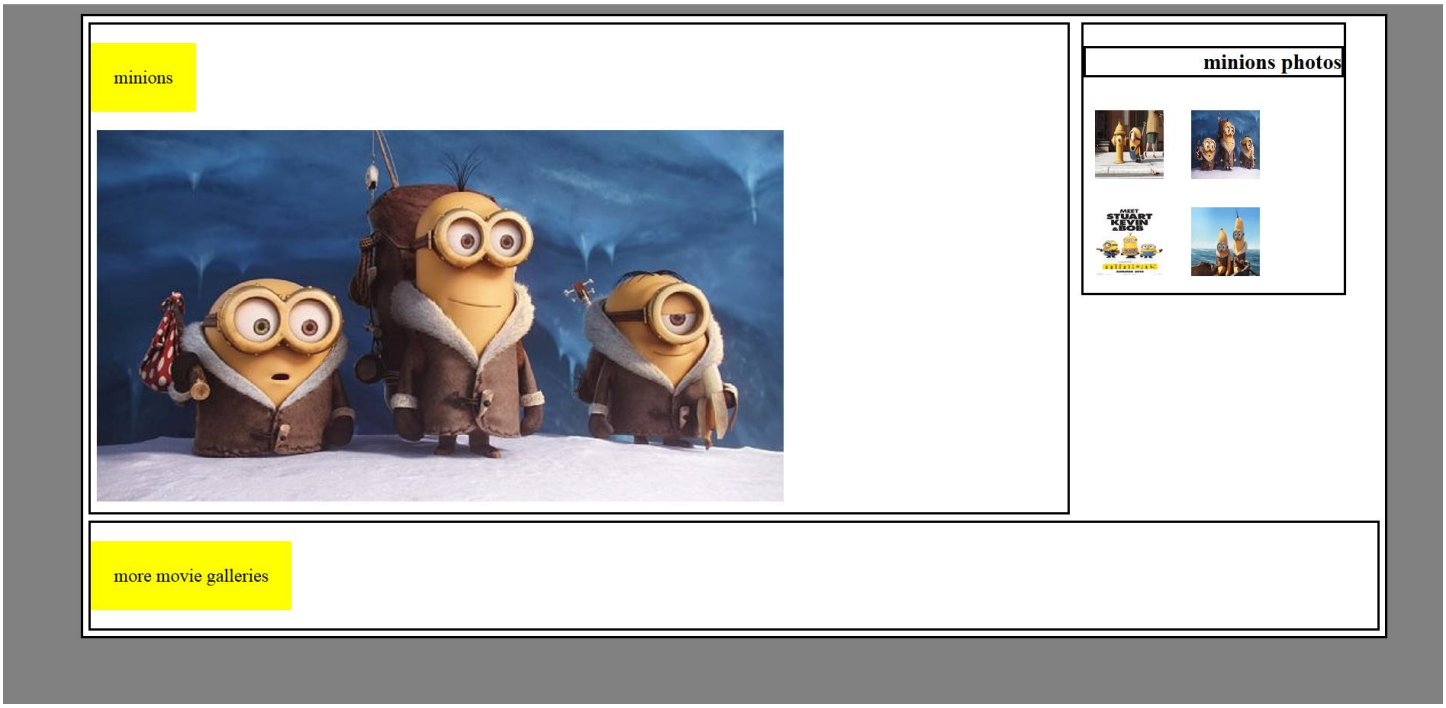
Good luck! You can do it!

('O')

Problem	Description	Earned	Max
1	CSS		
2	PHP		
3	PHP/JSON		
4	Ajax		
5	Regular Expressions		
6	SQL		
TOTAL	**Total Points**		**100**

1. CSS

Write the **CSS code** necessary to recreate the following appearance on-screen, exactly as shown.
The page uses the HTML code below. You are **not allowed to modify the HTML**.




```
<body>
  <div id="contentArea">
    <div id="bigPic">
      <p>minions</p>
      <div>
        <img src="minions.jpg" alt="big" />
      </div>
    </div>
    <div id="gallery">
      <h3>minions photos</h3>
      <img src="bydrant.jpg" alt="hydrant" />
      <img src="snow.jpg" alt="snow" />
      <img src="meet.jpg" alt="meet" />
      <img src="banana.jpg" alt="banana" />
    </div>
    <div id="more">
      <p>more movie galleries</p>
    </div>
  </div>
</body>
```

All borders shown are 2px thick, solid and black in color.

The elements that contain “minions” and “more movie galleries” have yellow backgrounds and 20px of space between the words and the edge of the background color.

There is 5px empty space around the outside of each div.

The small pictures are 60px wide and 60px tall and have 10px space around them. They should appear on the same line unless the line is full.

The div with the big picture and “minions” title is 75% of the width. The one with the small photos is 20%.

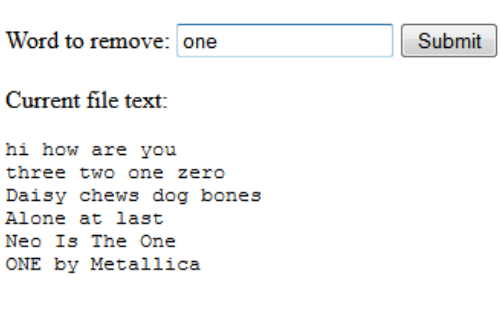
The background color of the page is gray but the background color of the area containing everything is white. The white area is 90% wide and centered.

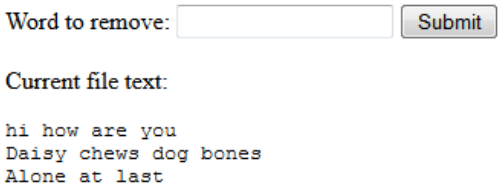
2. PHP

Write the PHP code for a web page `filter.php` that filters lines of text from a file. The page should contain a short form with a text box where the user can type a word. The page also displays the current contents of the file `text.txt` as a pre-formatted block. The form submits back to the same page, `filter.php`, as a POST request. When the word is submitted, your code should examine the contents of `text.txt` and remove any lines from the file that contain the given word, case-insensitively. Write the changes to the file so that any future viewings of the page will see the changes. You can write just the code dealing with the page's `body`; you don't need to output a head section or a complete page.

Match the exact word, not other words that contain it as a substring. For example, if the user submits the word "me" you would filter out lines containing the word "me", but not lines that just contain a word such as "men" or "game".

The following screenshots show the page as the user types the word "one" and after clicking Submit:





If the user makes a POST but somehow does not submit the query parameter for the word, or if the word they submit does not consist entirely of upper/lowercase letters, issue an HTTP 400 error and do not display the rest of the page. Use the browser's default styling; you do not need to write any CSS for this problem.

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"
               }
             ]
}
```

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:

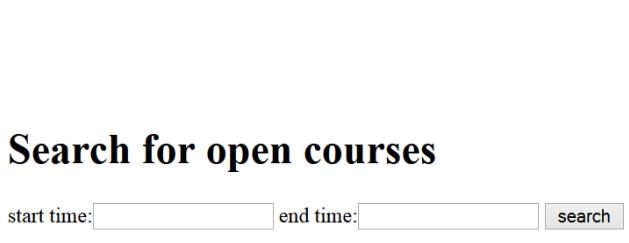
name **-** ***seats*** *seats*

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

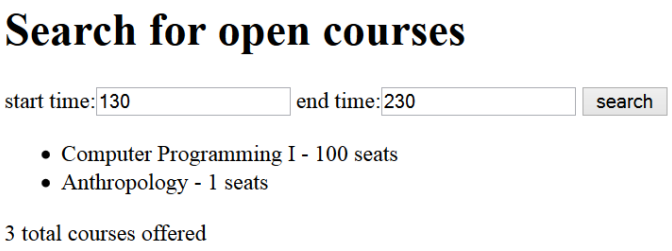
count *total courses offered*

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)

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

6. SQL

Write an SQL query on the `imdb` database that will return the names of all characters that appeared in two or more of the Pirates of the Caribbean movies; that is, movies whose name begin with the substring "Pirates of the Caribbean". Ensure that the results are returned in alphabetical order. If it helps you, you may assume that the character is played by the same actor in both movies.

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
...	

The following is a subset of the results returned:

```
+--------------+
| role         |
+--------------+
| Himself      |
| Marine       |
| Will Turner  |
+--------------+
```

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.

X. Extra credit

Draw a picture of your TA as a superhero.

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