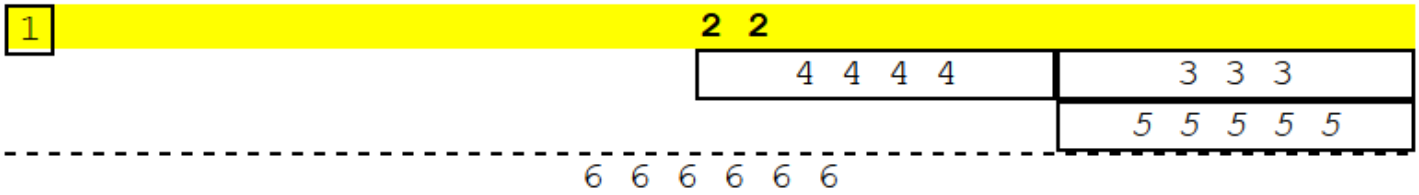
CSE 154 Final Review Section

1. CSS

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




```
<div>
  <span>1</span>
  <div id="div">2 2</div>
</div>
<span class="div">3 3 3</span>
<div>
  <div class="div">4 4 4 4</div>
  <div id="span">5 5 5 5 5</div>
  <div class="span">6 6 6 6 6 6</div>
</div>
```

All text now uses a monospace font at the default size.

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

The element with "2 2" now has a yellow background.

The elements with "3 3 3", "4 4 4 4", and "5 5 5 5 5" are now each exactly one fourth (1/4) of the page width.

The element "2 2" now has **bold** text, and the element "5 5 5 5 5" now has *italic* text.

2. PHP

Write the code for a web page `horoscope.php` and a page `horoscope-submit.php` which it submits to with a GET request. `horoscope.php` contains a small form that allows a user to type in the name of their sign. The form submits to `horoscope-submit.php` which displays a horoscope. This horoscope is the entire contents of a .txt file that is randomly chosen from a directory named the same thing as what the user typed in. For example, if the user typed in "taurus" your program might display the contents of "taurus/file1.txt". It should pick a file in the "taurus/" folder at random with equal probability. No matter what capitalization the user uses, you should look for a folder with a lowercase name.

Once a computer has accessed the site once, it should show anyone using the same computer the same horoscope for 24 hours. Individual computers where a user types in the same sign should get their own random horoscope. You should use cookies to implement this. If the user tries to access `horoscope.php` when they already have a chosen horoscope the page should redirect to `horoscope-submit.php`.

Once the user has successfully submitted the form once and gotten a horoscope and the 24 hours of redirecting are over, it should pre-fill the input box with the sign the user typed on the last access. The page should redirect for the first 24 hours after access and auto fill after that.

If the user types a sign that you do not have data for print an error message. The user should then be able to go back to `horoscope.php` and try again.

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

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"
               }
             ]
}
```

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:

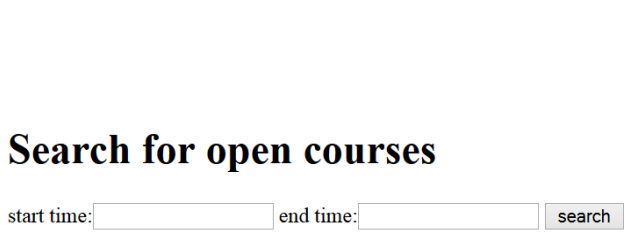
name **-** ***seats*** *seats*

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

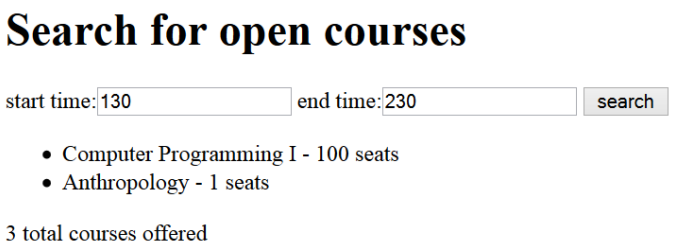
count *total courses offered*

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

6. SQL

The Springfield Elementary school board is trying to crack down on grade inflation. To do this, they are trying to figure out which teachers are giving a lot of high grades in their courses. They have asked you to write an SQL query that can be run on the `simpsons` database that will find the names and course names of all teachers who gave 2 or more grades of C- or better in a given course. The query should show the teacher's name and the course name.

Recall the `simpsons` database tables:

students

id	name	email
123	Bart	bart@fox.com
456	Milhouse	milhouse@fox.com
888	Lisa	lisa@fox.com
404	Ralph	ralph@fox.com

courses

id	name	teacher_id
10001	Computer Science 142	1234
10002	Computer Science 143	5678
10003	Computer Science 190M	9012
10004	Informatics 100	1234

grades

student_id	course_id	grade
123	10001	B-
123	10002	C
456	10001	B+
888	10002	A+
888	10003	A+
404	10004	D+

teachers

id	name
1234	Krabappel
5678	Hoover
9012	Stepp

The results of the query would be the following, because Krabappel taught 142 and gave Bart a B- and Milhouse a B+, and Hoover taught 143 and gave Bart a C and Lisa an A+:

```
+-----------+----------------------+
| Krabappel | Computer Science 142 |
| Hoover    | Computer Science 143 |
+-----------+----------------------+
```

If a teacher taught more than one course with >= 2 high grades, your query should only show that teacher's name once along with any one of the courses in which the high grades were given by that teacher. Recall that a grade above C- is a string that occurs earlier than the string 'C-' in alphabetical order.

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.