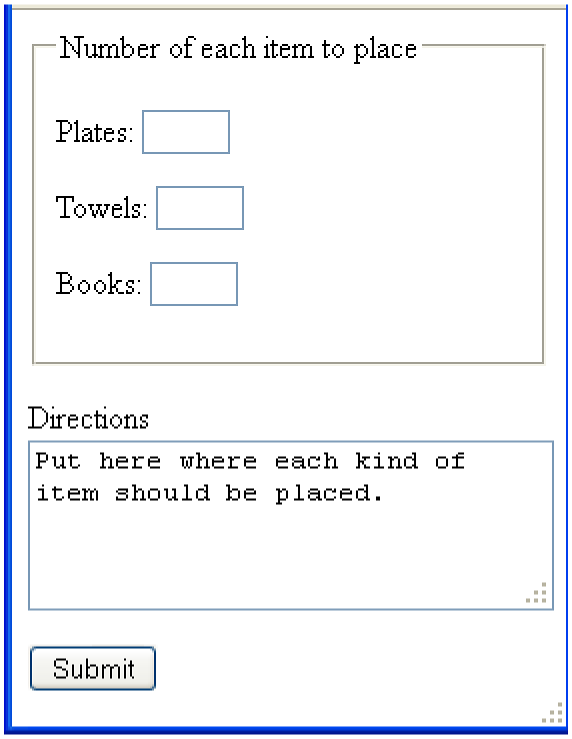
COMP 322 Internet Systems Spring 2019 Assignment 9

Due Tuesday, April 30 by 11:00 PM. 24 points total.

1 (5 pts.). Write an HTML document that is rendered like the screenshot below at left. It asks for the number of various items and has a textarea for directions on where to put them; the rendering shows the default text. The screenshot below at right shows the form filled out with numbers for the various items and directions for where to put them.



Number of each item to place

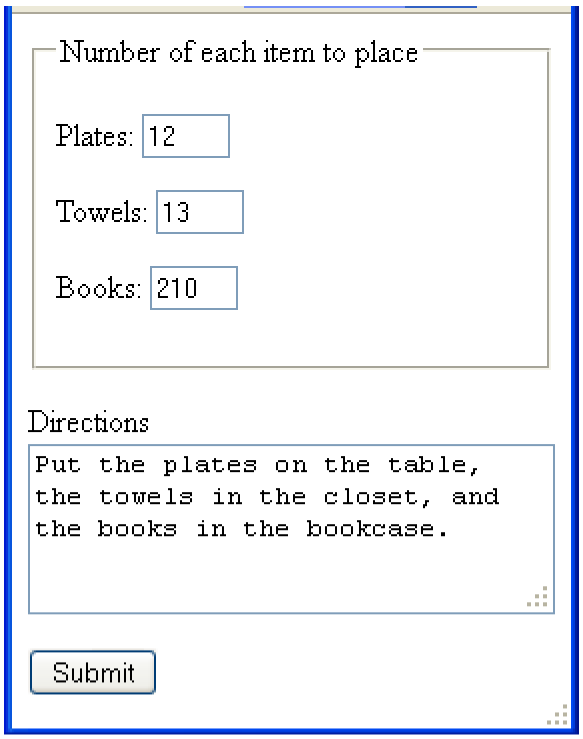
Plates:

Towels:

Books:

Directions

```
Put here where each kind of
item should be placed.
```



Number of each item to place

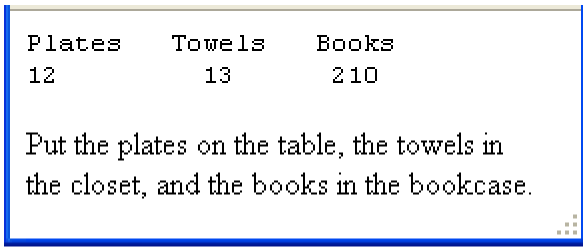
Plates: 12

Towels: 13

Books: 210

Directions

```
Put the plates on the table,
the towels in the closet, and
the books in the bookcase.
```

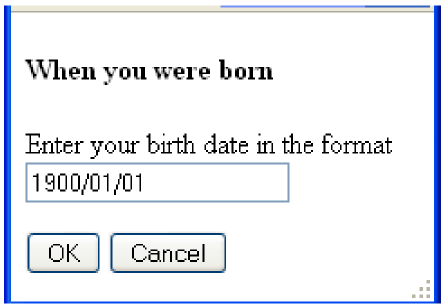


```
Plates    Towels    Books
12          13       210
```

Put the plates on the table, the towels in
the closet, and the books in the bookcase.

When the form is submitted, the request is sent to **prob1.php**. The rendering of the HTML it produces for the values shown above is captured in the screenshot at right. It echoes out headings for the various kinds of items and then uses **priintf()** to output the numbers below the headings. Reserve fields four characters wide for each field occupied by a number. (Each field is padded with spaces to bring its width up to four.) Have the leftmost number left justified in its field and the other two right justified. After the numbers, the directions are output. Split the directions so that they occupy two lines that are as close as possible to being of equal length without breaking up a word. (Hint: Set an index variable to the middle or just before the middle of the string and then increment it as long as the character it indexes is not equal to a space. Get rid of whitespace at the beginning or end of the directions string sent with the request by applying **trim()** to it.)

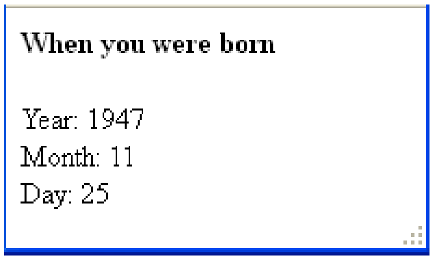
2 (7 pts.). Write an HTML document, **`prob2.html`**, that is rendered as shown in the screenshot at right. The example date in the textbox is the initial content. The button labeled OK is the submit button, and the button labeled Cancel is the reset button. When the form is submitted, a request is sent to **`prob2.php`**. The screenshot at right (lower) shows the rendering of the HTML it produces when **`1947/11/25`** is entered for the birth date. Use a regular expression to capture the three components of the date, which are then output on separate lines.



When you were born

Enter your birth date in the format

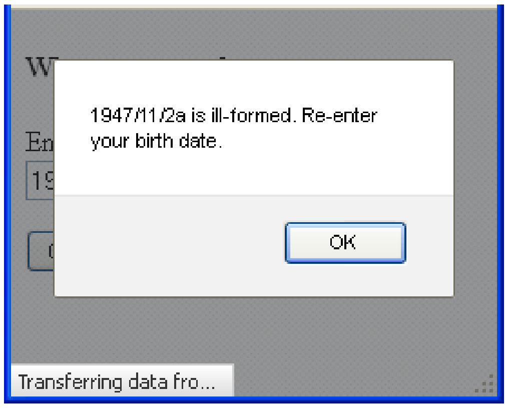
1900/01/01



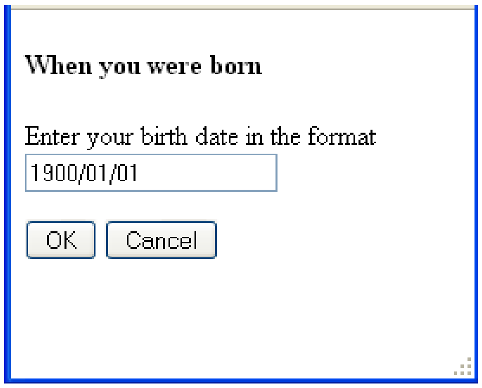
When you were born

Year: 1947
Month: 11
Day: 25

If the date is ill-formed, the HTML of the response raises an alert, as shown below at left. When the alert is dismissed, what is left is the same as the initial rendering of **`prob2.html`**. (The text in the lower left corner is produced by the system.) Copy and paste the form in **`prob2.html`** as pure HTML in an **`else`** clause in **`prob2.php`**. (The brackets, **`{`** and **`}`**, are in different PHP fragments.)



1947/11/2a is ill-formed. Re-enter your birth date.



When you were born

Enter your birth date in the format

1900/01/01

Produce the alert box using the device we are about to explain. We can embed JavaScript in the values of event attributes using the **`javascript`** pseudo-protocol—just prefix the JavaScript with "**`javascript:`**". For example, clicking the rendering of the following **`p`** element raises an alert box with content `Hi`.

```
<p onclick="javascript:alert('Hi')">Click me!</p>
```

We can also raise an alter box when the page finishes loading (as required for this problem) by having similar code as the value of the **`onload`** event attribute of the body:

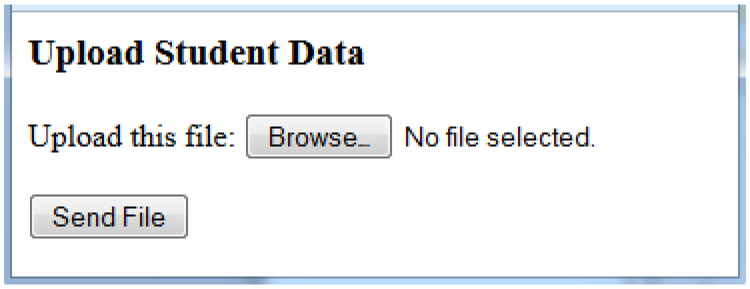
```
<body onload="javascript:alert('Hi')">
```

What makes doing this difficult in the current case is that the PHP has to construct the opening **`body`** tag with the **`alert()`** call with a string as its content, and this requires quotes within quotes, within quotes, and we have only two kinds of quotation marks (single and double). The solution is to escape one of the pairs of quotation marks. If **`$err`** contains the error message, we can use

```
echo " <body onload='javascript:alert(\"".$err."\");'>";
```

Don't include newlines in the error message: **`alert()`** does not allow multiline content.

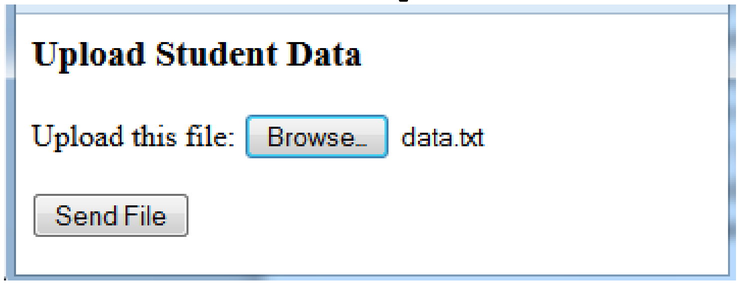
3 (6 pts.). Write an HTML document **`prob3.html`**, which is placed below your document root, and whose initial rendering is shown in the screenshot at right. The button labeled **Browse** is the rendering of an **`input`** element with **`type="file"`** and **`name="datafile"`**.



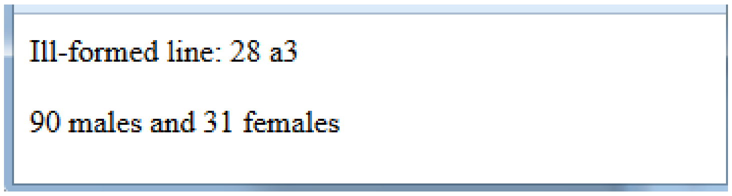
The Blackboard page for this assignment provides a file **`data.txt`**, whose content is as follows.

```
30 10
28 a3
29 12
31  9
```

These numbers are supposed to be the enrollments by gender in repeated offerings of a given course. In each line, the first number is the number of males, and the second number is the number of females. Note that the second line has an error: **`a3`** cannot be interpreted as a number. Place **`data.txt`** in the same folder as **`prob3.html`**. Click the **Browse** button and navigate to **`data.txt`** and select it, giving a rendering as in the screenshot at right. The button labeled **Send File** is the submit button. When it is clicked, a POST request with the uploaded file is sent to **`prob3.php`**, which is in the same folder as **`prob3.html`**.



Script **`prob3.php`** moves the data file to folder **`uploads`** immediately below the document root (i.e., the server sees this folder as **`"/uploads/"`**). It then opens this file and reads it. It sums the number of males and the number of females and includes the totals in the HTML of its response. Any lines in the file that cannot be interpreted as two numbers are echoed back, identified as ill-formed; they do not contribute to the totals. The screenshot at right shows the rendering of the response.



Ill-formed line: 28 a3

90 males and 31 females

Use a regular expression to extract the two numbers from a line read from the file. Allow any amount of whitespace (including none) before the first number and any amount of white space (including none) after the second number. Be sure to use the **`^`** and **`$`** anchors. There must be some whitespace (and nothing else) between the two numbers.

4 (6 pts.). For this problem, you will send (in a request) an array with string keys (i.e., an associative array). To see how this works, consider the following form, which submits a request to **ex.php** (in the same folder) when submitted. A rendering with values entered in the textboxes is shown at left below.

```
<form action="ex.php" method="get">
  <p>First number: <input type="text" name="num[first]" size="3" /></p>
  <p>Second number: <input type="text" name="num[second]" size="3" /></p>
  <p><input type="submit" value="Submit" /></p>
</form>
```

The names of the textboxes form an array with string keys. Note that the keys are not within additional quotation marks. When this form is submitted in my implementation, the URL in the address bar is

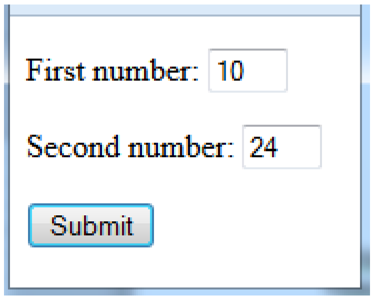
```
http://localhost/c322f13/HW9/ex.php?num[first]=10&num[second]=24
```

Script **ex.php** first assigns the value of form variable **num** (the array) to program variable **$num**.

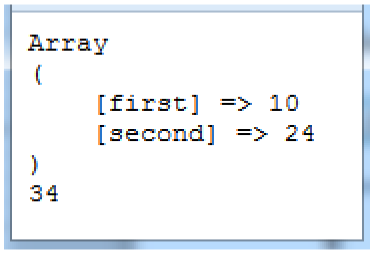
```
<?php
  $num = $_GET['num'];
?>
```

In the body of the HTML produced, it dumps out the array contents and echoes the sum of the two elements in the array. The rendering of the HTML produced is shown below at right.

```
<body>
  <pre>
<?php
  print_r($num);
  echo $num['first'] + $num['second'];
?>
  </pre>
</body>
```


First number: 10

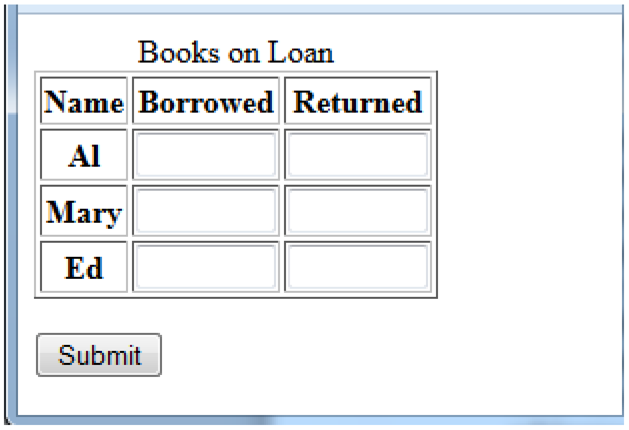
Second number: 24



```
Array
(
    [first] => 10
    [second] => 24
)
34
```

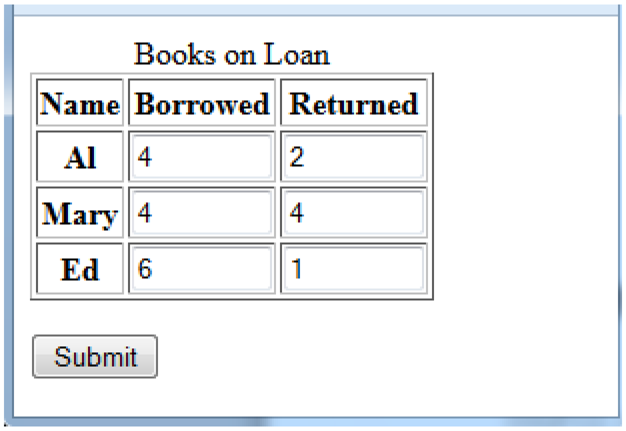
For this problem, write an HTML document **prob4.html**, initially rendered as shown in the screenshot at the top of the next page. This includes a table with a caption, a one-row head, and a three-row body; the row labels are **th** cells. It also includes a submit button; when it is clicked, a GET request is sent to **prob4.php** (in the same folder). The contents of the table-body cells except for the first column are textboxes with **size="8"**. The names in the textboxes in the second column form associative array **borrowed** whose keys are the same as the names in the first column, and the names in the textboxes in the third column form associative array **returned** whose keys again are the same as the names in the first column.

Script **`prob4.php`** reproduces the table with an additional column, labeled **Out** (for the number of books the person has checked out). The values in this column are the differences of the values in the previous two. The screenshot below right is the rendering of the HTML produced by **`prob4.php`** when the form in **`prob4.html`** was filled out as shown in the screenshot below left.



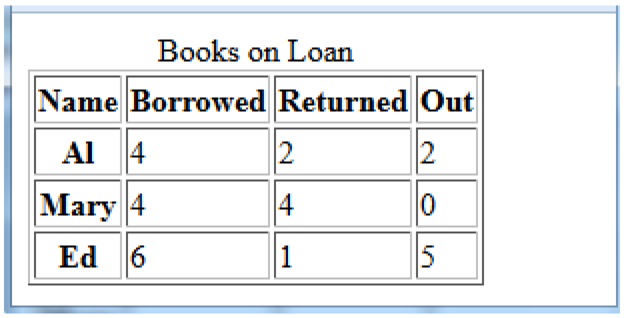
Books on Loan

Name	Borrowed	Returned
Al		
Mary		
Ed		



Books on Loan

Name	Borrowed	Returned
Al	4	2
Mary	4	4
Ed	6	1



Books on Loan

Name	Borrowed	Returned	Out
Al	4	2	2
Mary	4	4	0
Ed	6	1	5