PIC 40A: Homework 6 (due 11/20 at 10pm)

Like on previous homeworks, it is important that you meet the following requirements.

- You must upload your files to **Gradescope** before the deadline.
- You must upload your files to the **PIC server** in the appropriate directory before the deadline.
- Both submissions must be **identical** (down to the character).

  Never make changes to the PIC server submission after the deadline.

  (We can see when a file was last modified.)
- You must tell us (me and the grader) your **PIC username**.
- You must **validate** your HTML using https://validator.w3.org/.

  Ideally, with PHP, you should check that, after removing the PHP parts, the remaining HTML validates.

In this assignment you will submit three files...

1. `README.txt`. This will contain your PIC username.
2. `blog.php`, this is a php file but the live submission will look like a HTML file.
3. `post.php`, this is a php file but the live submission will look like a text file.

As mentioned above, you should submit all files to Gradescope before the deadline.
You should also submit the files to the PIC server. Save them in the directory

`/net/laguna/???...???/your_username/public_html/HW6`

(in the folder `HW6` within `public_html`). We should all be able to view your live webpage at

www.pic.ucla.edu/~your_username/HW6/

Now, I am just left to tell you what I want the webpage to do. See the next page!

New Page

You are going to make a page that allows users to make blog posts. A user will get to write in a box on the blog page. Their post will be stored in a text file on the host server and when the blog page is refreshed, their post will appear on the page. Newer posts will appear below older posts. In this assignment you'll use php to create a document that looks like a html file and one that looks like a text file.

Here's how they should work (see demonstration video)...

blog.php

This page will load formatted as if it were HTML. The page should include...

1. `<title>` can be set to "Our Posts".
2. There should be a heading within the header saying "Blog posts".
3. There should be a web `form` such that
    - It's `method` attribute is set to `POST`.
    - Inside the form there is a two `<label>` elements. The first label will say "Author: " and will be connected to a textbox. The second label will say "Content: " and be connect to a new HTML element called `<textarea>`.
    - By default, the textbox associated with the Author should be filled with the username of the person visiting the site (this is assuming they properly logged in via the login page).
    - Right of the `<textarea>` element those there will be an `<input>` element that will allow the user to submit the post.
4. There's a `<section>` below the form with the heading "Posts by other users:".
5. Once the form is submitted, the page should redirect to `post.php`.
6. Going back to this page will show the new post as part of a new paragraph in the section.
7. There should be a footer with copyright information.

post.php

The page will load formatted as a text document. That's the least interesting part of this page. The purpose of this page is to execute some code. You can land on this page two ways:

1. If you are redirected from submitting a post on the `blog.php` page. Then
    - The page will say "`post successfully written`"
    - If `posts.txt` did not exist before, it will now with the submitted information about the author and content.

- This submitted information will then be loaded when you visit `blog.php` by using php to inject into HTML. Text is in a new paragraph element and is formatted with the username bolded followed by " says: " and then the user's content.
- You might do your formatting on `blog.php`. Or you might chose to do some formatting before adding the information to `posts.txt`. You can assume blog posts can contain white space and line breaks. Users can type raw HTML.

  The weirdest post I'll test it with will be HW3 malicious666's post. Most websites would have something to 'sanitize' posts if they're going to be added to the page to prevent phishing or an awkward, lone opening/closing tag being injected.

2. If you go directly to the url where `post.php` exists, it will simply say

   ```
   Nobody has made a post.
   ```

   - If you go to the page for the first time ever (`posts.txt` doesn't exist) then `posts.txt` still won't exist.
   - If there's a `posts.txt`, then it remains unchanged.

Grading

1. (5 points) HTML in `blog.php` is as specified and page looks like it does in the demo.
2. (1 point) The HTML in `blog.php` validates (remove the php parts).
3. (1 point) The username from logging in fills the author textbox.
4. (1 point) If the user submits the post with the author textbox cleared, their username will be used by default.
5. (1 point) Submitting a post like "Hello, world!" adds the text to the page when its refreshed.
6. (1 point) Text is in a new paragraph element and is formatted with the username bolded followed by " says: " and then the user's content.
7. (1 point) Typing in a different author (different than the username) and adding some content will make a post with the different username.
8. (4 point) Posts should be able to handle the user's specific content formatting. Including them inserting HTML tags.
9. (1 point) `posts.txt` should keep a history of the user's posts.
10. (1 point) `posts.txt` should be deleted from the live submission so the grader can do all tests.
11. (1 point) `post.php` is formatted as a text file.
12. (1 point) Visiting `post.php` directly by the url does not create `posts.txt` if it doesn't exist.
13. (1 point) Visiting `post.php` directly by the url not change `posts.txt` if it does exist.