PIC 40A: Final project (due 3/20 at 5pm)

Like on all the 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.)
- For this last assignment, I'll let you submit as late as Friday 3/22 5pm with no penalty.
- You must tell us (me and the grader) your **PIC username**.
- You must **validate** your HTML using `https://validator.w3.org/`.

For the final project you will put together the web page you've been creating in the homework this quarter and add styling to it. It 80% of it will be graded based on the previous homework requirements so make sure you've made corrections. The last 20% will be based on the styling added. There are also some listed optional features you can add to your page. These will not be graded but it will be good practice for the in-person practical final.

In this assignment you will submit **seventeen(!)** files...

1. README.txt. This will contain your PIC username and any new features you want me to notice.
2. index.php
3. login.php, login.js, username.js, and h_password.txt.
4. logim.html (or logim.php), logim.js, scarf1.html, scarf2.html, and phish.js.
5. blog.php and post.php
6. merch.php, merch.js, and money.php

   These files will do all their meant to do and have the correct urls for any redirection.
7. style.css.

   This will add style to your pages.

You should also submit the files to the PIC server. Save them in the directory

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

You'll also need to create a folder called "sessions" with 755 permissions. (in the folder Final within public_html). We should all be able to test your live webpage at

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

Unfortunately, there has been issue with getting accounts to serve index.php. To have that page served you could include a index.html with the contents

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <script>window.onload = window.location.assign('index.php');</script>
  </head>
</html>
```

Now, I am just left to tell you what I want these files to achieve. See the next page!

Create a file called `style.css` and add a line saying `<link rel="stylesheet" href="style.css">` to the `<head>` element of `html` and `php` pages that are HTML formatted to style your webpage.

- (2 points) Use a **font** and **color scheme** different than Chrome's default;
- (1 point) use at least one **border** on each page;
- (1 point) use the same **text alignment** as in the demonstrational video.
- (2 points) All inline styling in the HTML pages should be removed.
  All styling should be in `style.css`

Other contributions...

- (2 points) Greeting on `index.php`
  Move your greeting to the upper right corner in the viewport. It will always remain there.
- (5 points) Navigation on `index.php`, `merch.php`, and `blog.php`.
  - Add a `<nav>` element to `<header>` of `index.php`, `merch.php`, and `blog.php`.
  - Have it navigate to the other pages using an anchor element.
  - Style it so that it always remains in the upper right of the viewport (leaving room for the greeting). When the page is zoomed at 100% it should not overlap `<main>`.
- (2 points) Content box on `blog.php`.
  Make the Content box the width of the container it's in and the height of `150px`.
- (5 points) Merch page
  - Make the credit appear at it's usually spot. Once you start scrolling past the point where it lives by default, it should remain on the page at top left for you to see.
  - Style the table with a margin and borders for the table and data. Make the pictures similar in size. Let the pictures have the effect that hovering over them makes them 70% opaque.
  - The receipt for checkout should come out in font that is monospaced.

For the final exam, you will be asked to create a new webpage for a client. Like the homework, you will be given a demonstration and specific criteria for grading. The webpage will be started and you will have to add the HTML, JS, PHP, and CSS to have it work like in the demo.

To give you some idea of what I could ask, here are some final practice prompts and optional features you might add to your webpage. To practice for the exam, plan out how you would implement these features. Then in one go, practice implementing at least one of these features.

On `blog.php`

- Have the newest posts show at the top rather than the bottom. (You can use another txt file to help or store the posts in a SQL database).
- Using an AJAX GET request, obtain the file with the blog posts every 4 seconds. You will need to create another file. Call it `post.js`.
- Using an AJAX POST request, send the information of the author and content to `post.php`. The page will remain on `blog.php` when the Post button is clicked (rather than refresh/redirect).

On `merch.php`

- Add a "sold out" feature. Record when a checkbox has been disabled so that it permanently remains disabled if the webpage is visited. You can store this information in a txt file or SQL database.
- Keep a SQL database of merchandise information including item name, item price, item description, and inventory quantity. Have the merch page dynamically fill up with the information from the database. If inventory quantity is 0, the checkbox is disabled.
- Make the credit-adding aspect of the page more private. Have the JS function `validate_coupon_code()` send out an AJAX POST request with the coupon box value to have a new php script `coupon.php` validate it. This would check a text file `coupons.txt` that would be added by the website owner. The file is formatted in the language of query strings with key-value pairs of the coupon code and associated credit: `COUPON5=5&&COUPON10=10&&COUPON20=20` and so on. If the coupon is listed, the credit would be updated on the page and in the SQL database for the user. Once a coupon is used, it would be removed from the text file.

On `index.php`

- Load some ascii art from a text file on the server into your page. As you add the text to your webpage, make it so you can add style to different elements of the ascii.

  For example in the crown below you might make all the parentheses gold, the double quotes bold, and the asterisks purple. Add a shadow to the left most thing that's not white space. You'll want to add to style.css.

```
     *
   *-|-*
   ..*..
 *"*****"*
```

```
" """"""" "
\*********/
(((((0)))))
```

- Using an AJAX get request to load the latest post from posts.txt. After 5 seconds load the next post in database. So the section at the bottom should have a paragraph is rotates through the posts.
- Add some JS to spy on the user and send the information back to the server. So either on the document or on a particular HTML element, record the location of their click and the time of their click. Add that to a database (txt or sql).

On `login.php`

- Have it so that if the user fails to login 5 times, the textbox and submit button are disabled for 15 minutes. Even if they refresh the page or close the browser and try to reopen it, the items will still remain disabled until 15 minutes are up. (Don't use incognito).
- Create a new login page that store a username and hashed password in a SQL database. Give the login page an additional link that redirects to a page with a web form which allows for new users with valid usernames to be added to the database.