CSCI 4350
Spring 2022
Midterm Exam

Last Name: ____________________

First Name: ____________________

You may use as an aid the following resources: A printout of any single homework assignment *you* have completed (HTML, CSS, Javascript/JQuery, PHP, SQL).

Write your name on this exam sheet and write your answers on the answer sheets provided.

1. (2 pts) What is the name of the web server software used by the csci server. (NOTE: The answer is *not* csci.hsutx.edu).

2. (3 pts) Briefly explain the primary differentiation between client-side web programming and server-side web programming.

3. (2 pts each) Suppose you have just logged in to your account on `csci.hsutx.edu`. Write the command(s) to accomplish each of these tasks:

   (a) change into your homework #3 directory

   (b) edit a file named `display.php`

   (c) view the last several lines of the PHP error log

4. Answer the following questions regarding password validation in PHP:

   (a) (2 pts) Suppose you have a site that allows only `https` on the page where users enter a password and that you store the passwords in a password-protected database. Would there be any reason in that case to use a one-way encryption scheme? Briefly explain.

   (b) (3 pts) Suppose a password has been been hashed using PHP's `password_hash` function. When a user logs why would it *not work* to do this comparison (assume the variables referred to have been properly initialized):

   ```
   if ($stored_password == password_hash($password_from_form)) ...
   ```

   (c) (2 pts) What would be the correct way to determine if the two passwords in problem 4b are a match?

5. (8 pts) Suppose you want a cookie-based login system that that expires after 30 minutes of inactivity. Write a PHP function named `checkEm` that is intended to be called at the top of every protected PHP page on your site. The function should do the following tasks:

   (a) check the login cookie to see if it has expired or not

   (b) if expired the user should be redirected to the login page (`login.html`)

   (c) if not expired the cookie's expiration should be reset to be 30 minutes in the future

   In case you forgot some basics about dealing with cookies in PHP, here is an example from the official PHP documentation:

   ```
   $value = 'something from somewhere';
   setcookie("TestCookie", $value, time()+3600);  // expires in 1 hour

   if (isset($_COOKIE['TestCookie'])) {  // Print an individual cookie
      echo $_COOKIE["TestCookie"];
   }
   ```

6. (4 pts) If you were going to carry out a CSRF attack on a web application, briefly describe how you would carry it out.

7. (6 pts) Write a PHP function called `prepSSN` that accepts a social security number as a parameter that comes from a client-side form. The SSN function should make sure it has the form: `###-##-####`. If not then the function should return `FALSE`. If it does have the right form then it should "normalize" the number so that the dashes are removed with only the digits remaining. The function should return the normalized value. HINT: use `preg_match(pattern, subject)` and `str_replace(search, replace, subject)`.

8. Suppose you want to implement a series of page counters for a site so that the page counters are kept in a PostgreSQL database on the local server. Suppose the database is named `counterdata` and is owned by a user `counteruser` with a password of `wheehaw`. Further suppose that the `public` schema contains a table named `counters` that has the following form:

```
CREATE TABLE counters (
   id          SERIAL,
   pagename    VARCHAR(30),
   value       INTEGER
);
```

NOTE: a `SERIAL` type is an integer that can be given a default value that increments a PostgreSQL sequence.

   (a) Suppose the following form (found in `create.php` is used to allow a user to create a new page counter in the database:

```
<form action="create.php" method="post">
   Page name: <input type="text" name="pagename" />
   Initial value: <input type="text" name="pagecount" />
   <input type="submit" />
</form>
```

   Write PHP statements that would go at the top of `create.php` that will validate the form data and display the form if visiting the form for the first time or if there is an error of some kind. As you do this be sure you handle these issues:

   i. (2 pts) If the current visit to the page is because of a POST request then verify the initial value provided in the form is a number.
   ii. (2 pts) Trim both inputs.
   iii. (8 pts) Check the database to ensure that no counter for that page already exists. If it does exist then you will skip further validation checks and go straight to the form at the bottom of the page.
   Example SQL command to determine whether a counter for a page exists:
   `SELECT COUNT(value) FROM counters WHERE pagename='fun.html'`
   This will create a table with a single row and a single column holding the number of counters for that page (0 or 1).
   iv. (4 pts) If the above checks are passed then insert a new counter in the database and set the value of the counter to the provided value from the form.
   Example SQL command to insert a new counter for a page:
   `INSERT INTO counter (pagename,value) VALUES ('fun.html',0)`

v. (4 pts) If a new page counter was created then you will, of course, need to continue to display the form at the bottom of the page. However, above the form display a message indicating which page name and initial value that were stored in database. (You can just put the words "FORM GOES HERE" rather than writing out the contents of the form.

NOTE: Your solution should properly handle SQL-injection concerns and XSS concerns.

(b) Suppose we have a page on our site for which we are keeping page count data. Rather than showing the page count on every page it has been decided that we will provide a button with a question mark on it. When the button is clicked we will use Ajax to extract the page's count from the database and replace the button's label with that count. Suppose the button on our page looks like this:

```
<button id="counter">?</button>
```

Further suppose that the server-side code for this functionality has already been written in a script named `getCounter.php`, which performs these basic actions:

```
look for a posted value named page_name
look up the page count associated with page_name in the db
echo the page count
```

i. (8 pts) Write the client-side jQuery code that will respond to the button click, generate an Ajax request, and update the button label to replace the question mark with the current page count. You will obtain the page count by calling the `getCounter.php` script described above. You will need to pass the current page's name to the server. That value is available with this Javascript command: `location.pathname.split("/").slice(-1)`

ii. (3 pts) If you had written `getCounter.php` would you need to be concerned about SQL injection given the fact that we are writing the Javascript code ourselves? Explain.

iii. (3 pts) In problem 8(b)i, we are putting data (i.e., the current page count) onto our page that came from the database. Do we need to be concerned about XSS attacks? Explain.

iv. (2 pts) In problem 8(b)i, do we need to be concerned about a CSRF attack?