CS 328 - Homework 8

Deadline

11:59 pm on **Friday, April 4, 2025**

Purpose

To get more practice with PHP postback documents that both create a form and response to that form, and to also practice some other PHP features; to practice more with CSS; and to write a PL/SQL stored procedure or stored function to be used on an upcoming application involving your "second" database.

How to submit

Each time you wish to submit your work-so-far, submit your files using **`~st10/328submit`** on nrs-projects, with a homework number of **`8`**.

Important notes

- **NOTE**: you are welcome to use **`require_once`**/**`include_once`**/**`require`**/**`include`** in your PHP documents!
  - BUT when you use them in homework problems' documents, be sure to also SUBMIT copies of all files that you are requiring/including!
- Remember: You are required to use **`normalize.css`** for all of your web pages for CS 328, and to add **`link`** elements for additional CSS external stylesheets *after* this (but still **within** the **`head`** element).

  EXCEPT for **`normalize.css`**, **DO NOT USE ANY CSS FRAMEWORKS or PREDEFINED LIBRARIES for this course** (unless you get prior, explicit permission). One of this course's purposes is to provide you with some practice with the basics of "plain" CSS, so you can better make use of frameworks and predefined libraries later.
- Remember: there are now **CS 328 PHP Coding Standards so far** posted on the public course web site, under References -- you are also expected to follow these for all course PHP documents.
- You are expected to follow all course coding standards posted on the public course web site; course documents are also expected to validate as "strict"-style HTML, and valid CSS.

Problem 1

FUN FACT: Here is PHP's most basic `while` loop:

```
<?php
    while (desired_expr)
    {
        statement;
        ...
        statement;
```

```
    }
?>
```

Its semantics should be reasonably familiar -- when the `while` is reached, its `desired_expr` is evaluated. If it is something PHP considers "truthy", the statements in its loop body are executed. And, at the end of the loop body, `desired_expr` is evaluated again, and the loop body entered again if it is "truthy" -- and so on, until the `desired_expr` evaluates to something PHP considers "falsey" when checked after a loop body execution, at which point the loop is exited.

The purpose of this problem is to give you practice writing a postback PHP document (that **both** creates a form **and** crafts a response to that form when it is submitted, as seen in the posted `328lect09-2.php` and your Week 9 Lab Exercise's `328lab09.php`), and that **also** happens to include a **`while`** loop.

Using the posted **`html-template.html`** as the initial basis, create a PHP document **`328hw8-1.php`** that meets the following requirements:

- Include your name and last modified date in its opening comment, **AND** the **URL** this can be run from.
  - (You *will* lose some credit if this URL does not work when I or the grader paste it into a browser!)
- **Within** the **`head`** element, edit the **`title`** element, giving it appropriate content.
- **Within** the **`head`** element, after the `link` element specifying that this document is being styled by `normalize.css`, add a second **`link`** element following course style standards specifying that this document will then be further styled using a file **`328hw8-1.css`** that you are about to create for this problem.
  - **Do not include any inline or internal CSS rules in your `328hw8-1.php`.**
- Within the **`body`** element, include an **`h1`** element with appropriate content.
- **Within** the **`footer`** element near the end of the **`body`** element, add a **`p`** element whose content includes your name.
- Your **`328hw8-1.php`**'s logic should be designed so that its response includes either a form, or a response to its form -- its response should never include both.
- The initial **`form`** element generated by your **`328hw8-1.php`** should have an **`action`** attribute whose action is:

  `"<?= htmlentities($_SERVER["PHP_SELF"], ENT_QUOTES) ?>"`

  ...and should contain at least:
  - one number-field (an **`input`** element with **`type="number"`**)
  - **at least ONE OTHER form widget** of your choice
  - a submit button (an **`input`** element with **`type="submit"`**)
- When this form is submitted, the response generated by your **`328hw8-1.php`** should VISIBLY do something, based on the "other" form widget(s), the **number of times** entered into the number-field.

  - (This could be as simple as, for example, outputting something entered into a textfield the number of times entered into the number-field, or the response could be something more involved -- you get to decide!)
  - The response should **ALSO** include an **`a`**/anchor element (hypertext link) with appropriate text that links back to your **`328hw8-1.php`**.
  - Make sure your **`328hw8-1.php`** appropriately **sanitizes** ALL of the inputs provided by the user when this form is submitted! (And remember that this care is necessary **regardless** of the form control in use -- the actual request might NOT be coming from your form!)

Your **`328hw8-1.css`** should meet the following requirements:

- Include a comment including at least its file name, your name, and the last-modified date.
- Include rules that "nicely" (tastefully, attractively, and readably) lay out and format this form and the response to this form.
- If you change any of the default foreground and background colors, make sure that, for any text atop a background, the **contrast** between their colors is **at least WCAG 2 AA Compliant** based on the tester at: https://snook.ca/technical/colour_contrast/colour.html.
- If your **`form`** does not happen to include a **`fieldset`** element, then include rule(s) adding a visible, attractive **border** to at least your **`form`** element.
- Include rule(s) that will result in your **`form`** ***not*** taking up the entire width of the body, and being **centered** within the body.
- Include rule(s) that will result in the **submit button** somehow being **centered** within the **`form`**.
- Add rule(s) using either **CSS flexbox layout** or **CSS grid layout**, your choice, to noticeably layout something within either your **`form`** or a container element within your form.
- Add additional CSS rules as desired to further attractively format elements of these documents.
- Make sure your resulting **`328hw8-1.css`** validates as valid CSS.

Strict-validate the two parts generated by your **`328hw8-1.php`** as you did for the Week 9 Lab Exercise's `328lab09.php`:

- Put your **`328hw8-1.php`**'s URL in a browser and **view its source**, **copy and paste** that **source** into a file named **`328hw8-1-1.xhtml`**, and put the URL of your **`328hw8-1-1.xhtml`** into the validator.
- Put your **`328hw8-1.php`**'s URL in a browser **and fill out and submit its form**, *then* **view that *response*'s source**, and **copy and paste** that ***response*'s source** into a file named **`328hw8-1-2.xhtml`**, and put the URL of your **`328hw8-1-2.xhtml`** into the validator.

Submit your resulting files:

- **`328hw8-1.php`** (and all additional files it uses, if any)
- **`328hw8-1-1.xhtml`** and **`328hw8-1-2.xhtml`**
- **`328hw8-1.css`**.

FUN FACT: HTML checkbox elements and PHP, Part 1

What you should already know about checkbox elements:

When a checkbox form widget such as the one in this fragment:

```
<input type="checkbox" name="game_desired" id="game_yes" />
<label for="game_yes">Check if you want to play a game</label>
```

...is within a form and that form is submitted, then:

- If that checkbox **IS** checked when the form is submitted, it sends the name=value pair:

  `game_desired=on`

  ...for that checkbox.

  - And, if that form had **`method="post"`**, then in a PHP is handling that submission, **`$_POST["game_desired"]`** would have the value **`"on"`**.
- If that checkbox is **NOT** checked when the form is submitted, it sends **NO** name=value pair for that checkbox.

Problem 2

The purpose of this problem is to give you a chance to try out PHP's **`array_key_exists`** function.

You can see **IF** an array **KEY** exists in a PHP associative array using:

`array_key_exists(``$desired_key, $desired_array`**`)`**

For example, for:

```
$my_first_assoc_array = ["a" => 1, "b" => 2, "c" => 3];
```

these would be true:

- **`array_key_exists("a", $my_first_assoc_array) === true`**
- **`array_key_exists("j", $my_first_assoc_array) === false`**

For this problem, you are going to create yet another postback PHP document ((that **both** creates a form **and** crafts a response to that form when it is submitted, as seen in the posted `328lect09-2.php` and your Week 9 Lab Exercise's `328lab09.php`), that also uses PHP's **`array_key_exists`** function.

Using the posted **`html-template.html`** as the initial basis, create a PHP document **`328hw8-2.php`** that meets the following requirements:

- Include your name and last modified date in its opening comment, **AND** the **URL** this can be run from.
  - (You *will* lose some credit if this URL does not work when I or the grader paste it into a browser!)
- **Within** the **`head`** element, edit the **`title`** element, giving it appropriate content.
- **Within** the **`head`** element, after the `link` element specifying that this document is being styled by

`normalize.css`, add a second **`link`** element following course style standards specifying that this document will then be further styled using a file **`328hw8-2.css`** that you are about to create for this problem.

  - **Do not include any inline or internal CSS rules in your `328hw8-2.php`.**

- Within the **`body`** element, include an **`h1`** element with appropriate content.
- **Within** the **`footer`** element near the end of the **`body`** element, add a **`p`** element whose content includes your name.
- Your **`328hw8-2.php`**'s logic should be designed so that its response includes either a form, or a response to its form -- its response should never include both.
- The initial **`form`** element generated by your **`328hw8-2.php`** should have an **`action`** attribute whose action is:

  `"<?= htmlentities($_SERVER["PHP_SELF"], ENT_QUOTES) ?>"`

  ...and should contain at least:

  - a **fieldset** with **at least four checkboxes**

    each of which has a ***DIFFERENT*** **`name`** attribute,

    (and each with a logically-associated **`label`** element)

    - You get to choose the "theme" for these checkboxes -- you need to use one that is different from any of the posted class examples.

  - a submit button (an **`input`** element with **`type="submit"`**)

- When this form is submitted, the response generated by your **`328hw8-2.php`** should include the following:

  - If **at least one** of the checkboxes was checked,

    - it should display a **`ul`** element with an **`li`** element for each selected checkbox.

    and if **none** of the checkboxes was checked,

    - it should display a **`p`** element whose content is a statement that none were selected.

  - You are expected to appropriately use the function **`array_key_exists`** in this part of your PHP document.
  - The response should **ALSO** include an **`a`**/anchor element (hypertext link) with appropriate text that links back to your **`328hw8-2.php`**.
  - Make sure your **`328hw8-2.php`** appropriately **sanitizes** ALL of the inputs provided by the user when this form is submitted! (Yes, even from checkboxes -- the actual request might NOT be coming from your form!)

Your **`328hw8-2.css`** should meet the following requirements:

- Include a comment including at least its file name, your name, and the last-modified date.
- Include rules that "nicely" (tastefully, attractively, and readably) lay out and format this form and the response to this form.

- If you change any of the default foreground and background colors, make sure that, for any text atop a background, the **contrast** between their colors is **at least WCAG 2 AA Compliant** based on the tester at: https://snook.ca/technical/colour_contrast/colour.html.
- Include rule(s) that will result in your **`form`** ***not*** taking up the entire width of the body, and being **centered** within the body.
- Include rule(s) that will result in the **submit button** somehow being **centered** within the **`form`**.
- Add rule(s) using either **CSS flexbox layout** or **CSS grid layout**, your choice, to noticeably layout the **fieldset** containing your **checkboxes**.
- Add additional CSS rules as desired to further attractively format elements of these documents.
- Make sure your resulting **`328hw8-2.css`** validates as valid CSS.

Strict-validate the two parts generated by your **`328hw8-2.php`** as you did for the Week 9 Lab Exercise's `328lab09.php`:

- Put your **`328hw8-2.php`**'s URL in a browser and **view its source**, **copy and paste** that **source** into a file named **`328hw8-2-1.xhtml`**, and put the URL of your **`328hw8-2-1.xhtml`** into the validator.
- Put your **`328hw8-2.php`**'s URL in a browser **and fill out and submit its form**, *then* **view that *response*'s source**, and **copy and paste** that ***response*'s source** into a file named **`328hw8-2-2.xhtml`**, and put the URL of your **`328hw8-2-2.xhtml`** into the validator.

Submit your resulting files:

- **`328hw8-2.php`** (and all additional files it uses, if any)
- **`328hw8-2-1.xhtml`** and **`328hw8-2-2.xhtml`**
- **`328hw8-2.css`**.

FUN FACT: HTML checkbox elements and PHP, Part 2

Demonstrated in zyBooks Chapter 5 - Section 5.11 - Participation Activity 5.11.7:

IF:

- you have several "logically grouped" checkboxes
- you give them ALL the SAME **`name`** attribute that ends with **`[]`**
- you give them EACH a *different* **`value`** attribute with a value representing that checkbox's meaning

THEN:

- If at least ONE of those checkboxes is checked when the form is submitted,

  it sends a name=value pair where:

  - the name is that common **`name`** attribute's value **without** the **`[]`**,
  - and the value is an ***array*** containing the **`value`** attribute values for EACH checkbox that was

checked at the time of submission!

- If **NONE** are checked, ***no*** name=value pair is sent.

That is, consider this fragment:

```
<input type="checkbox" name="transport[]" value="bicycle"
       id="bike" />
<label for="bike"> Bicycle </label> <br />

<input type="checkbox" name="transport[]" value="motorcar"
       id="car" />
<label for="car"> Car </label> <br />

<input type="checkbox" name="transport[]" value="skatebd"
       id="board" />
<label for="board"> Skateboard </label>
```

If this fragment is within a form with **`method="post"`** and that form is submitted, then:

- If, for example, the checkboxes for **Bicycle** and **Skateboard** are checked when the form is submitted,

  then **`$_POST["transport"]`** will contain an array whose values are **`["bicycle", "skatebd"]`**

- If **NONE** of these checkboxes with **`name="transport[]"`** is checked when the form is submitted, it sends **NO** name=value pair for any of those checkboxes.

Well, this needs to be tried out!

Problem 3

The purpose of this problem is to try out **"logically grouped" checkboxe**s and how PHP can respond to a form containing them.

For this problem, you are going to create yet another postback PHP document ((that **both** creates a form **and** crafts a response to that form when it is submitted, as seen in the posted `328lect09-2.php` and your Week 9 Lab Exercise's `328lab09.php`).

Using the posted **`html-template.html`** as the initial basis, create a PHP document **`328hw8-3.php`** that meets the following requirements:

- Include your name and last modified date in its opening comment, **AND** the **URL** this can be run from.
  - (You *will* lose some credit if this URL does not work when I or the grader paste it into a browser!)
- **Within** the **`head`** element, edit the **`title`** element, giving it appropriate content.
- **Within** the **`head`** element, after the `link` element specifying that this document is being styled by `normalize.css`, add a second **`link`** element following course style standards specifying that this document will then be further styled using a file **`328hw8-3.css`** that you are about to create for this problem.

  - **Do not include any inline or internal CSS rules in your `328hw8-3.php`.**

- Within the **`body`** element, include an **`h1`** element with appropriate content.
- **Within** the **`footer`** element near the end of the **`body`** element, add a **`p`** element whose content includes your name.
- Your **`328hw8-3.php`**'s logic should be designed so that its response includes either a form, or a response to its form -- its response should never include both.
- The initial **`form`** element generated by your **`328hw8-3.php`** should have an **`action`** attribute whose action is:

  `"<?= htmlentities($_SERVER["PHP_SELF"], ENT_QUOTES) ?>"`

  ...and should contain at least:

  - a **fieldset** with **at least four checkboxes**

    each of which has the ***SAME*** **`name`** attribute whose name ends with **`[]`**,

    each of which has a **DIFFERENT** **`value`** attribute whose name logically describes the choice

    (and each with a logically-associated **`label`** element)

    - It is fine if these have the same "theme" as those in Problem 2, but make sure you implement them as described above, sot hat you practice with this kind of "logically grouped" checkboxes' approach.

  - a submit button (an **`input`** element with **`type="submit"`**)

- When this form is submitted, the response generated by your **`328hw8-3.php`** should include the following:

  - If **at least one** of the checkboxes was checked,

    - it should display a **`ul`** element with an **`li`** element for each selected checkbox.

    and if **none** of the checkboxes was checked,

    - it should display a **`p`** element whose content is a statement that none were selected.

  - You are expected to appropriately use the function **`array_key_exists`** in this part of your PHP document.
  - The response should **ALSO** include an **`a`**/anchor element (hypertext link) with appropriate text that links back to your **`328hw8-3.php`**.
  - Make sure your **`328hw8-3.php`** appropriately **sanitizes** ALL of the inputs provided by the user when this form is submitted! (Yes, even from checkboxes -- the actual request might NOT be coming from your form!)

Your **`328hw8-3.css`** should meet the following requirements:

- Include a comment including at least its file name, your name, and the last-modified date.
- Include rules that "nicely" (tastefully, attractively, and readably) lay out and format this form and the response to this form.

- If you change any of the default foreground and background colors, make sure that, for any text atop a background, the **contrast** between their colors is **at least WCAG 2 AA Compliant** based on the tester at: https://snook.ca/technical/colour_contrast/colour.html.
- Include rule(s) that will result in your **`form`** ***not*** taking up the entire width of the body, and being **centered** within the body.
- Include rule(s) that will result in the **submit button** somehow being **centered** within the **`form`**.
- Add rule(s) using either **CSS flexbox layout** or **CSS grid layout**, your choice, to noticeably layout the **fieldset** containing your **checkboxes**.
- Add additional CSS rules as desired to further attractively format elements of these documents.
- Make sure your resulting **`328hw8-3.css`** validates as valid CSS.

Strict-validate the two parts generated by your **`328hw8-3.php`** as you did for the Week 9 Lab Exercise's `328lab09.php`:

- Put your **`328hw8-3.php`**'s URL in a browser and **view its source**, **copy and paste** that **source** into a file named **`328hw8-3-1.xhtml`**, and put the URL of your **`328hw8-3-1.xhtml`** into the validator.
- Put your **`328hw8-3.php`**'s URL in a browser **and fill out and submit its form**, *then* **view that *response*'s source**, and **copy and paste** that ***response*'s source** into a file named **`328hw8-3-2.xhtml`**, and put the URL of your **`328hw8-3-2.xhtml`** into the validator.

Submit your resulting files:

- **`328hw8-3.php`** (and all additional files it uses, if any)
- **`328hw8-3-1.xhtml`** and **`328hw8-3-2.xhtml`**
- **`328hw8-3.css`**.

Problem 4 - a PL/SQL stored function or procedure for your "second" database

Consider your selected "second" database, the one you specified as part of **Homework 3 - Problem 1**.

As foretold in Homework 7, for this problem you are to decide on and implement either a **PL/SQL stored function** or a **PL/SQL stored procedure** using your selected "second" database.

What might be a useful PL/SQL stored function or stored procedure for your "second" database? It does not have to be complex, but it has to be potentially useful, and you will be calling it from the application tier as part of a future homework problem.

For it to be feasible to call your PL/SQL stored function or stored procedure from a postback PHP document, your PL/SQL stored function or stored procedure needs to meet these requirements:

- It needs to involve table(s) from your selected "second" database.
- It needs to expect **at least one parameter** (to be able to practice with a particular important security-related feature we will be discussing).
- It needs to **NOT** depend on `dbms_output.put_line` calls (since their output cannot reach the

application tier).

- It should be potentially useful to the audience for and/or to an application programmer developing applications atop your selected "second" database.

It does not have to be complex, as long as it might be potentially useful. For example:

- A PL/SQL stored function might conveniently compute and return a count, sum, average, maximum, or minimum computation based on a choice selected by an end-user.
- A PL/SQL stored procedure might insert, update, or delete into/from a table based on information entered into a form by an end-user.

Create a **SQL script** in a file whose name includes **`second-db`** that meets the following requirements:

- Include comment(s) containing at least your name, **`CS 328 - Homework 8`**, and the last-modified date.
- Include a SQL*Plus **`spool`** command to spool the results of running this SQL script to a file whose name includes **`second-db-out`** and whose suffix is **`.txt`**,

  followed by a **`prompt`** command including **your name**.
- (Be sure to **`spool off`** at the end of this script, after your statements for creating and testing your stored function or stored procedure.)
- Following the CS 328 style standards, design and implement a potentially-useful PL/SQL stored function or stored procedure for your selected "second" database that meets the requirements listed above.
- Follow that definition with **at least two testing calls** that demonstrate that it works.
  - Be sure to print to the screen the expected results for each test, followed by its actual results.
  - For a stored procedure, you also will likely need to include one or more **`select`** statements showing that the desired side-effect(s) occurred.
  - If your tests will modify your "second" database, precede the set-up for those tests with a **`commit;`** statement and follow them with a **`rollback;`** statement.

Submit your files **`*second-db*.sql`** and **`*second-db-out*.txt`**.

(IMPORTANT NOTE:

- I or the grader should be able to run your latest-submitted `*design.sql` and `*populate.sql`, and then your `*second-db*.sql`, successfully.
  - SO: **IF** you make any changes to your `*design.sql` or `*populate.sql`,

    then make sure you ALSO:
    - submit updated copies of your `*design.sql` or `*populate.sql` (using a homework number of **33**)
    - along with your `*second-db*.sql` and `*second-db-out*.txt` (submitted using a homework number of **8**)!