

Welcome to Coders Assembly’ IS113 Code Clinic exercises. There are a total of 13 questions, with various levels of difficulty. You are strongly encouraged to solve **ALL the 1\* and 2\* questions first**.

Please try not to use Google, try to simulate as if you are doing these questions in Lab Test settings.

**Download resource folder:**

<https://drive.google.com/drive/folders/1J8ey58z1K43b7-64BZgPFf0RmGXmkY1g?usp=sharing>

As you code, try to **time yourself** to solve it as fast as possible. The following timings are suggested for each difficulty:

| **One Star [\*]** | 10 minutes or less |
| --- | --- |
| **Two Star [\*\*]** | 10 to 20 minutes (depending on difficulty) |
| **Three Star [\*\*\*]** | 20+ minutes, until your Lab Test time is up.  If you spend more than 30 minutes and are unable to solve / start, skip the question. |

Feel free to ask the instructor or your mentors for help if you are stuck. You can also move on to other questions first.

If you need any clarifications regarding the questions, you can telegram me @StepUpUrGame.

**--------------------------------------------------------------------------------------------------------------------------**

###### **Q1: Tell me a Joke!**

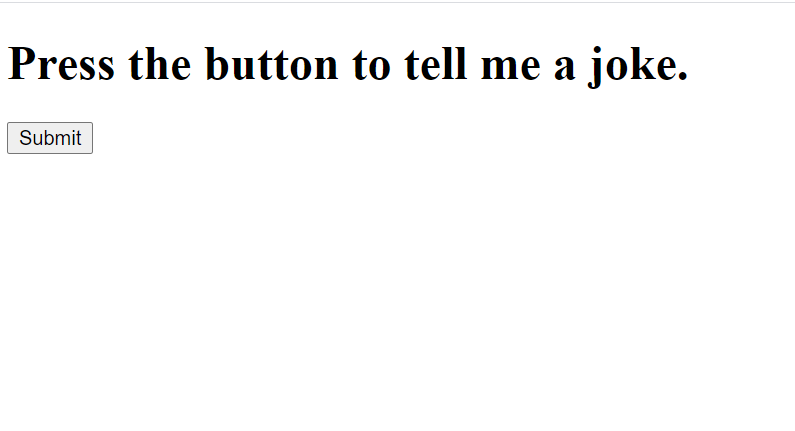
**---------------------------------------------------------------------------------------------------------------------------**

[\*] In q1.php, you are given an multi-dimensional associative array ($jokeArr), and a function getRandomIndex(). Read the comment to understand what the function does.

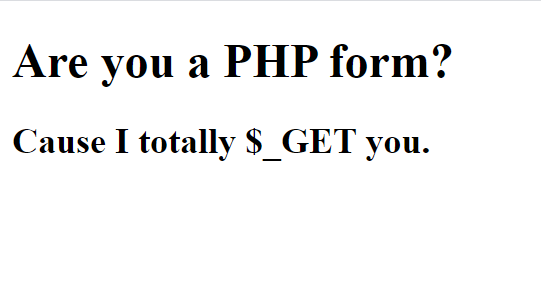
Edit q1.php such that when the submit button in q1.html is clicked, q1.php will display a random joke and its answer. The joke should be in a **<h1> header**, while the answer should be in a **<h2> header**. DO NOT hardcode the answers.

Your output should look like this:

(q1.html before the submit button is pressed)



(After submit button is pressed; redirected to q1.php)

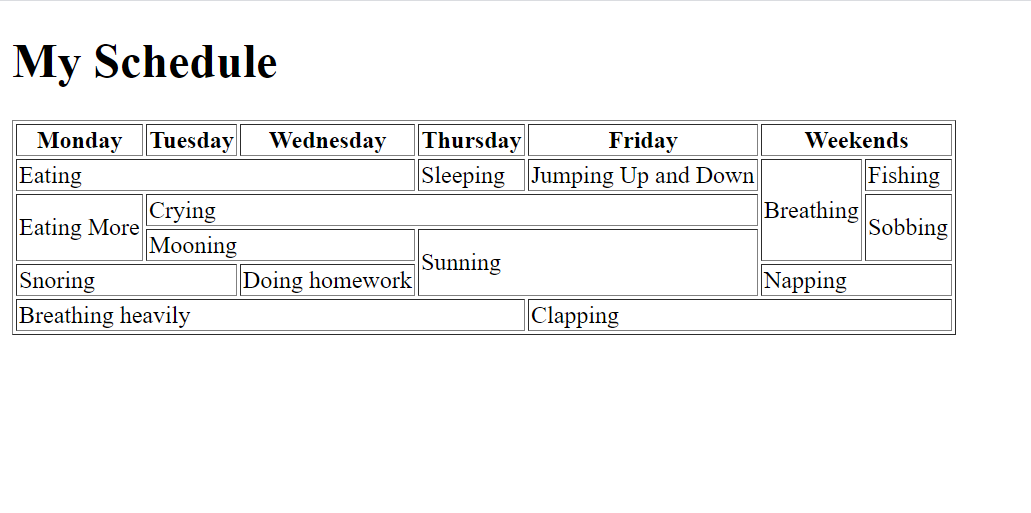


**--------------------------------------------------------------------------------------------------------------------------**

###### **Q2: My Terrible Schedule**

**---------------------------------------------------------------------------------------------------------------------------**

[\*] Create a schedule on your *q2.html*. The header above the table should be <h1>. Your q2.html should look exactly like the below:



**--------------------------------------------------------------------------------------------------------------------------**

###### **Q3: My Terrible Schedule, with a dash of PHP**

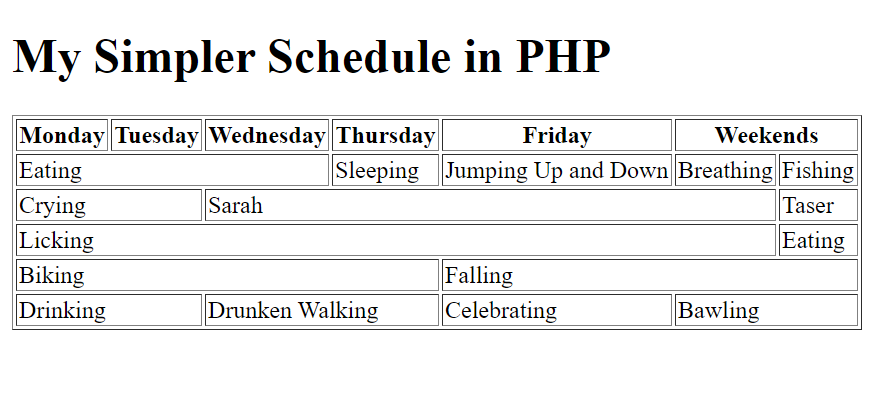
**---------------------------------------------------------------------------------------------------------------------------**

[\*\*] You are given an array $scheduleArr. Each element in the array is an associative array.

Using this array, populate the table in *q3.php* with data from the array. You are **NOT** allowed to hardcode values.

Hint: You should use foreach or for loops to help you out!

If everything works well, you should have the following table:



**--------------------------------------------------------------------------------------------------------------------------**

###### **Q4: Lottery Number**

**---------------------------------------------------------------------------------------------------------------------------**

[\*\*] You are given a file called q4.html, DO NOT MODIFY THIS FILE.

You are given a multi-dimensional array ($prizeArr) in q4.php. You are expected to do form validation; if the input submitted is **NOT a valid positive integer**, it should print a h1 header telling the user to enter a valid positive integer.

If a positive integer is entered, the page should print the prizes that have been won if there is a matching number, as shown in $prizeArr.

If the number does not match, a h1 header is printed telling the user that they did not win anything.

Take a look at the following test cases:

| **Before Form Submission** | **After Form Submission** |
| --- | --- |
| Or |  |
|  |  |
|  |  |

**--------------------------------------------------------------------------------------------------------------------------**

###### **Q5: Calculator**

**---------------------------------------------------------------------------------------------------------------------------**

[\*] In *q5.html*, edit the <form> such that when the “Calculate” button is clicked, the form is submitted to *q5.php* via HTTP GET.

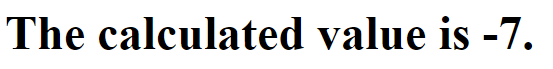
In q5.php, add your code to calculate, based on the first and second number and the operator. The answer should be printed as a **<h1> header**.

Check your code with the following test cases.

When first number is **2**, second number is **3** and operand is **+** :



When first number is **3**, second number is **10** and operand is **-** :



When first number is **5**, second number is **5** and operand is **\*** :



When first number is **8**, second number is **5** and operand is **/** :



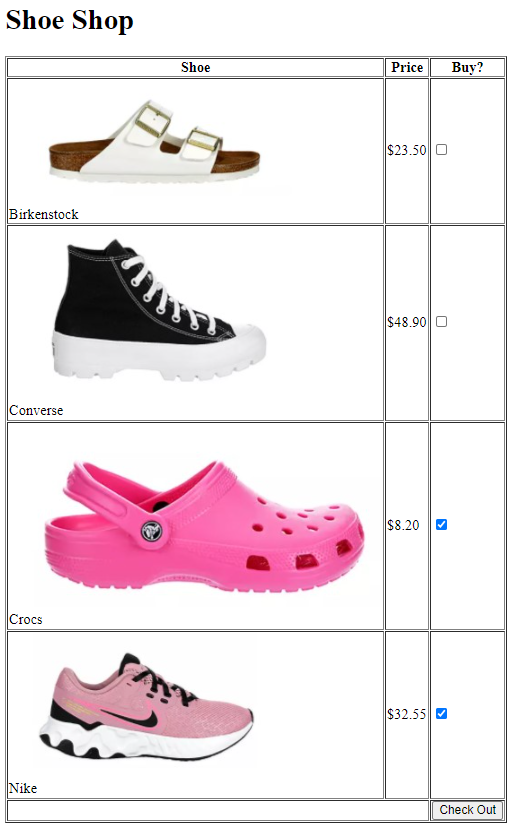
**--------------------------------------------------------------------------------------------------------------------------**

###### **Q6: Shopping for Shoes**

**---------------------------------------------------------------------------------------------------------------------------**

[\*] PART A

Complete q6.html such that it displays the following page:

****

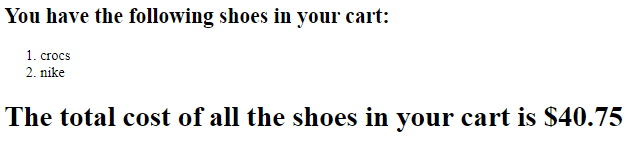
[\*] PART B

Complete q6.php, such that when the form is submitted to it via **HTTP GET**, q6.php will display the selected shoes in an ordered list, and calculate the total cost of the shoes.

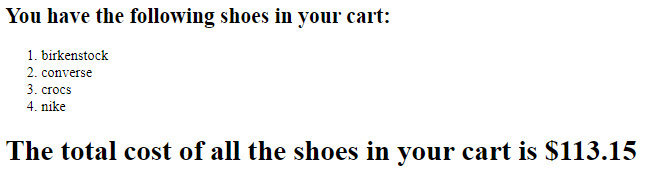
You should create an associative array to help you.

NOTE: The first header is <h2>, while the second header is <h1>

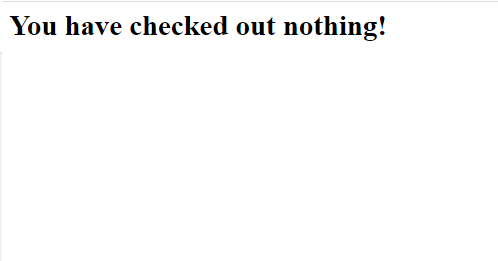
TEST 1: (When Crocs and Nike are selected and Check Out button is pressed)



TEST 2: (When all shoes are selected and Check Out button is pressed)



TEST 3: (When no shoes are selected and Check Out button is pressed)



**--------------------------------------------------------------------------------------------------------------------------**

###### **Q7: Generate a Ninja (WARNING: Got Some Naruto Shippuden Spoilers)**

**---------------------------------------------------------------------------------------------------------------------------**

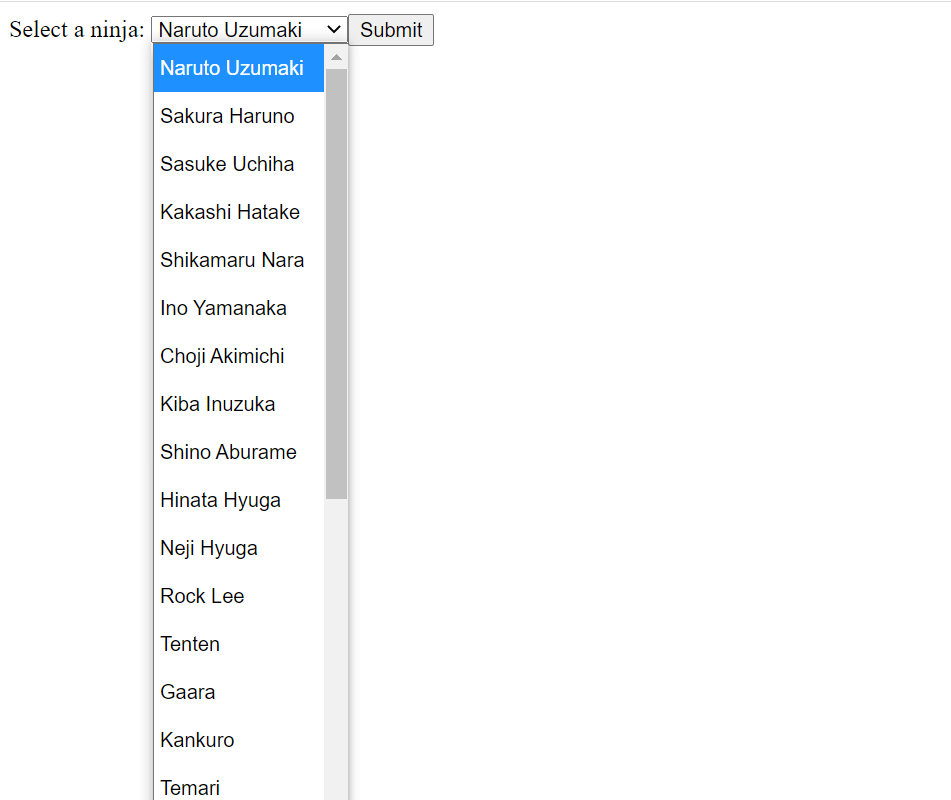
[\*] PART A

You are given an array $ninjaArr. Each element consists of a string, consisting of the name, village, gender, description, status and signature jutsu of a ninja, separated by commas ( , ). In some strings, there is an additional attribute “Deserted”, located at the end of the string.

Create a <select> dropdown menu displaying the list of names of each ninja in the $ninjaArr. **You should use PHP to dynamically fill in the options in the dropdown menu. DO NOT HARDCODE THE NAMES**.

Hint: Explore the use of the explode() function.

The page should look like this:



(This menu appears when the dropdown is clicked).

[\*\*] PART B

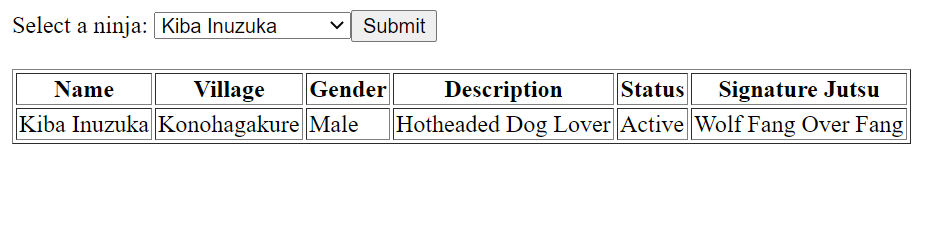
Edit q7.php such that when the Submit button is clicked, a table consisting of the selected ninja’s details is displayed below.

If “Deserted” is present in the ninja’s string, another row should be printed with the words in red. (Hint, use <font> tags to help you with this).

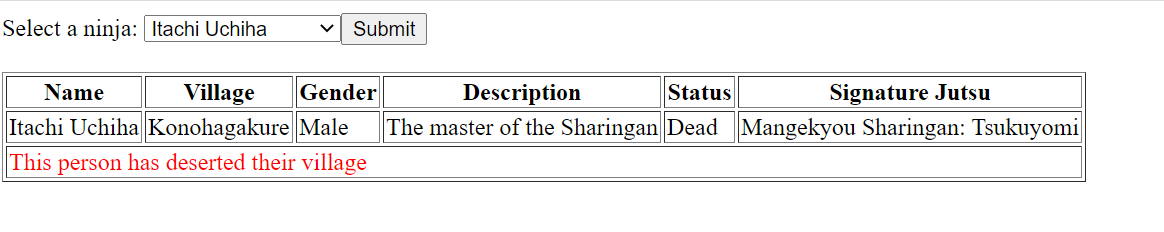
Upon clicking submit, the selected option should also be

Your output should look like the following:

TEST 1 (When Kiba Inuzuka is selected and the Submit button is clicked):



TEST 2 (When Itachi Uchiha is selected and Submit button is clicked):



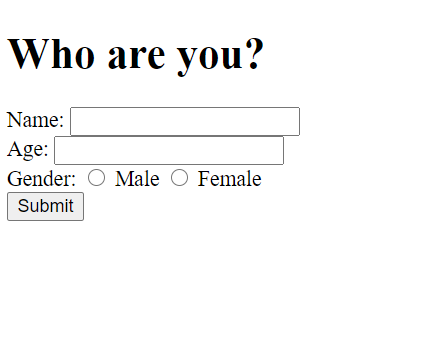
**--------------------------------------------------------------------------------------------------------------------------**

###### **Q8: Introduce Yourself**

**---------------------------------------------------------------------------------------------------------------------------**

[\*] PART A

Edit q8.html such that it creates the following page:



Take note of the following:

* When the form is submitted, it should redirect the user to q8.php
* When the form is submitted, the data input should NOT appear in the URL bar.
* You should use the ‘number’ input type for Age.

[\*\*] PART B

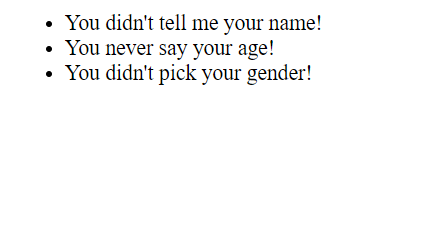
Edit q8.php, such that it takes in the user’s gender, name and age and creates a welcome message.

If the person is above 30 years old, the message will include “Wow, you are so OLD!”. If not, the message should include “So young ah!”

If the person does not enter their name, age or gender, error messages will be shown in the form of an **unordered list**.

You may use the following test cases:

TEST 1 (When nothing is filled and the form is submitted):



TEST 2:

| **Before Form Submission** | **After Form Submission** |
| --- | --- |
|  |  |

TEST 3:

| **Before Form Submission** | **After Form Submission** |
| --- | --- |
|  |  |

TEST 4:

| **Before Form Submission** | **After Form Submission** |
| --- | --- |
|  |  |

TEST 5:

| **Before Form Submission** | **After Form Submission** |
| --- | --- |
|  |  |

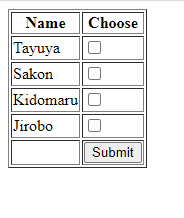
**--------------------------------------------------------------------------------------------------------------------------**

###### **Q9: Sound Four Tic Tac Toe**

**---------------------------------------------------------------------------------------------------------------------------**

[\*\*] PART A

Edit q9.php to display the following table in the webpage.



Take note of the following details:

1. When the submit button is pressed, the form should **self-submit** to itself via **HTTP GET**.
2. After the submit button is pressed, checkboxes that were selected should R**EMAIN ticked** after form submission. (Hint: Use the ‘checked’ attribute of checkboxes)
3. The form data should be displayed **dynamically** ( NO HTML HARDCODING). You should create an array and make use of it to generate your table data.

[\*\*\*] PART B

When the form is submitted, if less than 2 checkboxes are ticked, a message “Please select at least 2 people!” should be printed as a <h1> header.

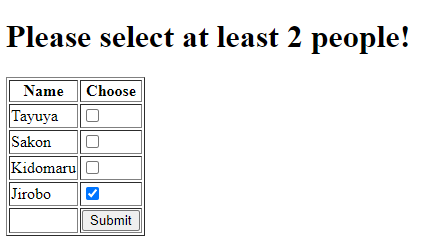
If 2 or more checkboxes are ticked, a 3x3 table should be displayed. In each square, an image of one of the ticked characters should be shown. The image generated should be at **RANDOM** (use the given function generateRandomIndex to help you with this).

In addition, if there is a row, column or diagonal that contains 3 images of the same kind, a <h1> header should be displayed for each type (row, column and diagonal) that is seen.

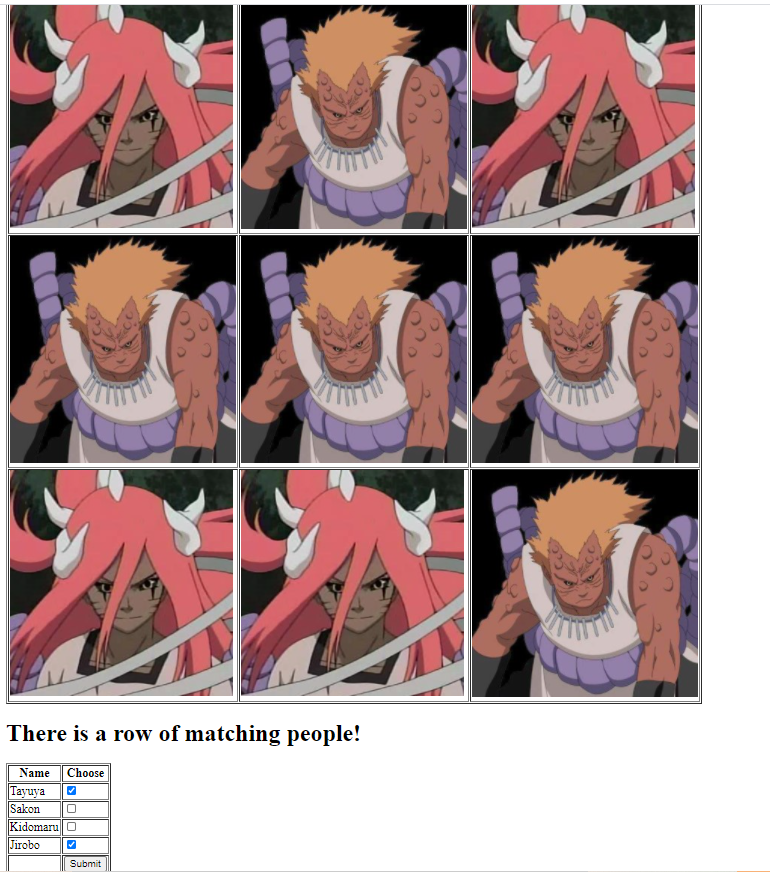
For example, if there are 2 columns and 1 row with 3 matching images, two <h1> headers should be displayed; one for the column and one for the row.

You may refer to the following test cases:

TEST 1 (If less than 2 checkboxes are ticked and submit button is pressed):



TEST 2 (If Jirobo and Tayuya are ticked and there is a matching row):



TEST 3 (If Tayuya and Sakon are ticked, and there is matching columns and diagonals):



TEST 4 (If all 4 members are ticked, and there is no matching row, column or diagonal):



**--------------------------------------------------------------------------------------------------------------------------**

###### **Q10: Bidding**

**---------------------------------------------------------------------------------------------------------------------------**

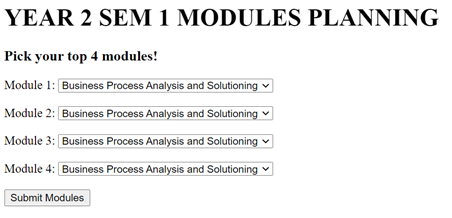
\*\* This question was created by Jonathan Wong

Go to bidding directory. Complete Part A and B inside bidding.php and Part C inside bidding\_partc.php.

[\*] PART A

You are given **bidding.html** which has 4 drop-down lists with different modules as their options. There is an error in bidding.html, pls change action to “bidding.php” @ line 12.

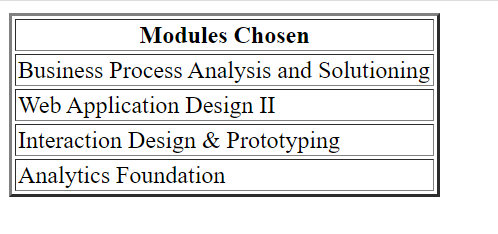
Complete **bidding.php** file so that it displays the following message when the user selects and submits repeated modules:

  
**TEST 1 (If repeated modules were selected):**



(“Back to form” should bring user back to bidding.html)

**TEST 2 (If no repeated modules were selected):**

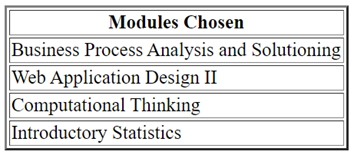


( The order of the table is based on the module selection; from top to bottom )

[\*\*] PART B

Modify **bidding.php** such that when the user submits 4 different modules, the page shows the selected modules in a table with a border of size 2.

You are ENCOURAGED to come up with your own associative array.



[\*\*\*] PART C

Complete **bidding\_partc.php**. Reuse codes from previous parts whenever possible.

You are given an associative array in **bidding\_partc.php**.

$modules\_info = [

"Business Process Analysis and Solutioning" => [

["Rafael J. BARROS", "KE Ping Fan", "Swapna GOTTIPATI", "A. M. Aditya"],

["LUM Eng Kit", "Fiona LEE", "CHUA Hong Ngoh", "CHUA Hong Ngoh"],

1

], …

];

When a user sends the form from **bidding\_partc.html**, use this array to help populate a table that shows the professors and instructors teaching the respective selected modules.

Concurrently, calculate and display below the table, the total credit units that this user will be clearing in this coming semester.

Things to take note:

· All headings will use heading size 1.

· Border of the table will take size 1.

· For the value of the units cleared, use heading size 3.

· Ensure that your code can still handle a simple form validation just like in Part B.

· Ensure that your code is dynamic, i.e. do NOT assume that for each module, the number of professors teaching is always the same.

**--------------------------------------------------------------------------------------------------------------------------**

###### **Q11: Fire your Chefs!**

**---------------------------------------------------------------------------------------------------------------------------**

[\*\*] PART A

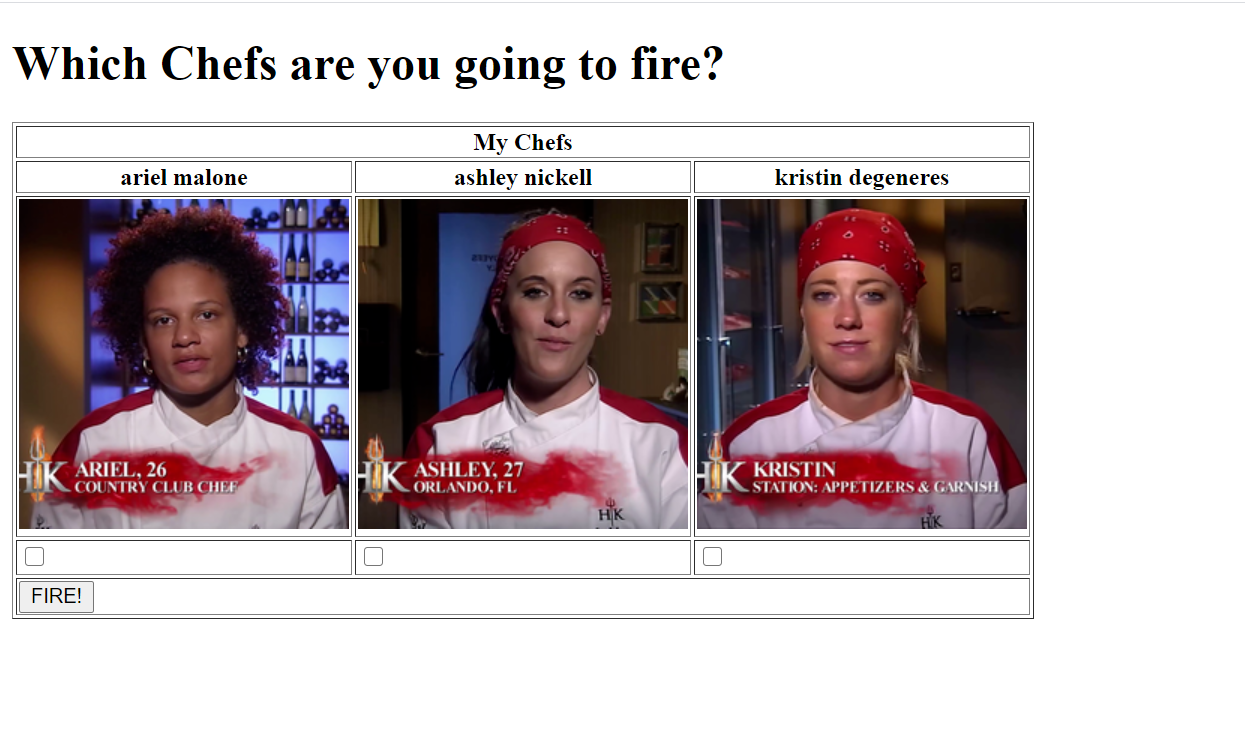
You are given q11.php, which contains an array called $chefArr. In this array, each key represents the last name of the chef, while the values represent the first names of the chef.

You are also given 3 images (in .png) in a folder called img.

Complete q11.php, such that when it is opened, it displays a form with a table featuring the chefs’ full name, their picture, as well as a checkbox.

The table should be DYNAMICALLY filled using PHP and the $chefArr given. DO NOT hardcode.

When done, the page should display this when it is opened on your browser:



See next page for Part B.

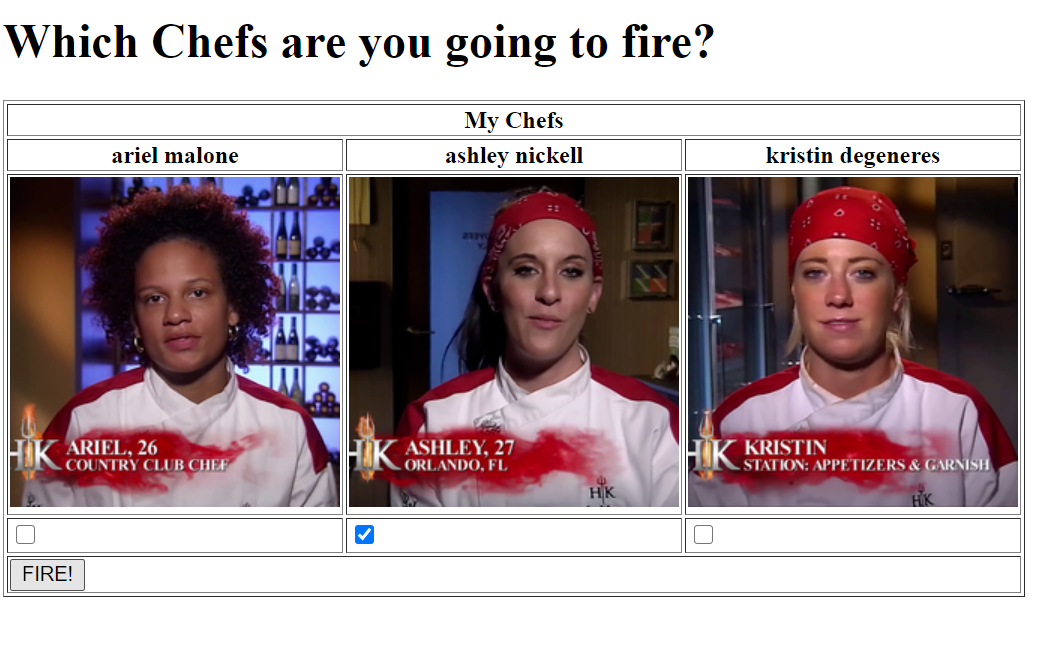
[\*\*\*] PART B

In q11.php, add PHP code such that when the form is submitted (via HTTP POST), the chefs whose checkboxes are ticked will be removed from the table. If all chefs are fired, a message will be placed in the table and the form submission button is removed.

*NOTE: The form is a self-submitting form, meaning that it will submit to itself, instead of another file.*

Here are some sample outputs:

TEST 1: Before pressing FIRE! (ashley nickell is ticked)



TEST 1: After pressing FIRE!



TEST 2: After pressing FIRE! (ariel malone and ashley nickell are ticked)



TEST 3: After pressing FIRE! (all chefs are ticked)



NOTE: Before each test case, please REFRESH the page.

**--------------------------------------------------------------------------------------------------------------------------**

###### **Q12: Travel**

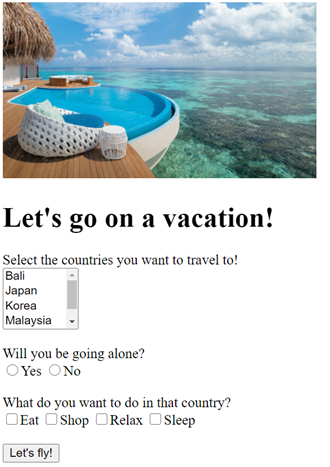
**---------------------------------------------------------------------------------------------------------------------------**

\*\* This question was created by Jonathan Wong

Go to travel directory. Complete Part A in **travel\_a.html** andParts B, C, D in **travel\_bcd.php** respectively.

[\*] PART A

Complete the HTML form inside **travel\_parta.html**. You are to replicate the form shown in the screenshot below.



The text “Let’s go on a vacation!” will use heading size 1 and the image displayed is named **maldives.jpg** and has a height of 200. (Use height attribute in <img> tag)

Ensure that when the user clicks on the words of the radio and checkbox options, that option will be selected. (Use label tags)

[\*] PART B

What you have completed in Part A is a static template.

Complete **travel\_partbcd.php** such that the countries, yes/no options and activities in the form are now dynamically displayed.

You may copy your code from Part A and add the necessary codes to fulfil the requirements. You may also come up with your own arrays.

[\*\*\*] PART C

Modify **travel\_partbcd.php** so that it performs the following form validation:

| **travel\_partbcd.php (user input)** | **travel\_partbcd.php (after form submission)** |
| --- | --- |
| // NOTE that the **first time** this page loads, it must show the form as shown above.  // The error messages must NOT be shown. |  |
|  |  |
|  |  |
|  |  |

**\*Note that after the user submits the form, the selected options remain selected and the necessary error messages are shown below.**

[\*\*] PART D

When the user successfully submits the form with NO errors, the following should be displayed:



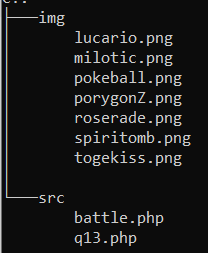
The message between the table and ordered list uses heading size 3 and the image width is 200 (Use height attribute in <img> tag).

**--------------------------------------------------------------------------------------------------------------------------**

###### **Q13: Pokemon Selection**

**---------------------------------------------------------------------------------------------------------------------------**

In q13 folder, your directory is arranged as shown:



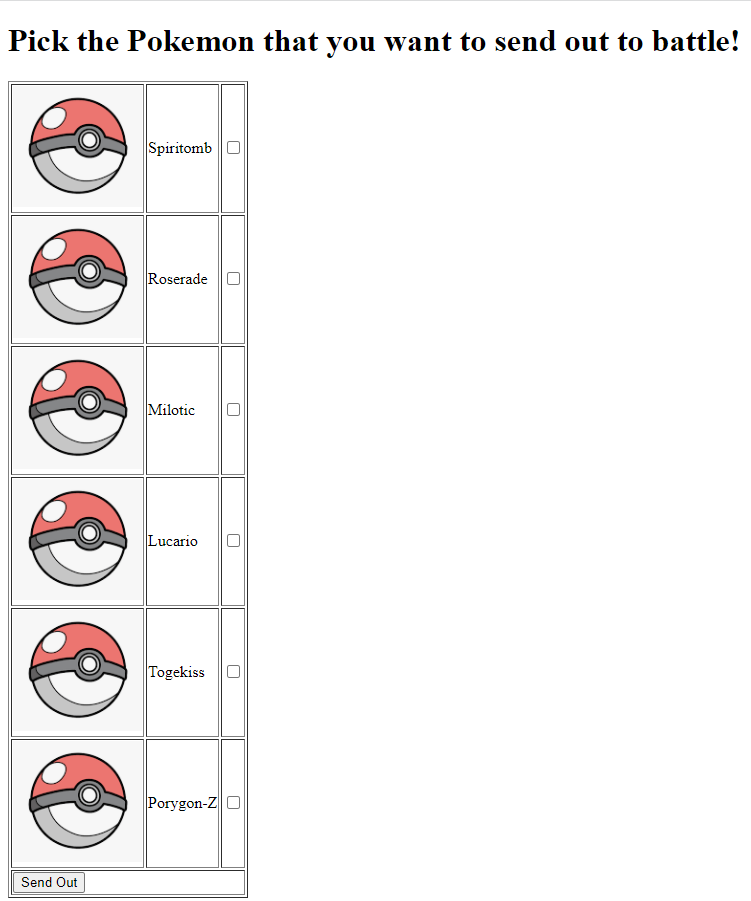
[\*] PART A

In q13.php, you are given an associative array called $pokemon\_arr, which represents 6 pokemon with their names as the array key, and their image path as the value. As of now, the image paths are labelled with ‘ADD\_IMAGE\_PATH\_HERE’.

Replace these values with the proper image path of the respective pokemon. For example, “Milotic” should link to the image “milotic.png”.

[\*\*] PART B

Complete the table in q13.php such that it displays the following page as shown in the next page:



Take note of the following:

* The form should submit to battle.php via HTTP POST (done for you)
* Each table row should contain the **image of a Pokeball (pokeball.png)**, followed by the **name of the pokemon** and a **checkbox that contains the image path of the respective pokemon’s image**.
* You MUST generate this table dynamically with PHP using $pokemon\_arr. Do NOT hardcode.
* The last row of the table should have a “Send Out” button, which will take the user to the next page.

See next page for Part C

[\*\*] PART C

Complete battle.php. When the user submits the form in q13.php, the page should show images of the pokemon that the user has chosen to send out.

If no pokemon were selected, the form should remain on q13.php, even after pressing the “Send Out” button. (HINT: Use header() to help you)

There should only be a maximum of 3 pokemon in one row. If there are more than 3 pokemon selected, the 4th pokemon onwards should be placed in another row.  
  
Your code should meet the following 4 test cases:

TEST 1:

| **Before Form Submission** | **After Form Submission** |
| --- | --- |
| * When no pokemon are selected and the “Send Out” button is pressed. | The page remains at q13.php |

TEST 2:

| **Before Form Submission** | **After Form Submission** |
| --- | --- |
| * When Roserade and Lucario are selected and the “Send Out” button is pressed. | Note: Use a <h1> header. |

TEST 3:

| **Before Form Submission** | **After Form Submission** |
| --- | --- |
| * When Spiritomb, Milotic, Togekiss and Porygon-Z are selected and the “Send Out” button is pressed. | Note: Use a <h1> header. |

TEST 4:

| **Before Form Submission** | **After Form Submission** |
| --- | --- |
| * When all pokemon are selected and the “Send Out” button is pressed. | Note: Use a <h1> header. |