CIS 332 Week 3 Assignment

In this assignment we will practice how to work with Associative Arrays and Arrays of Arrays (2D arrays). Both associative arrays and 2D arrays are important when we work with databases. The 2D arrays emulate the table structure of a database. The associative array emulates a single row of a database table. Therefore, a 2D array can be expressed as a collection of associative arrays (or rows). For example, In this assignment, we will get practice working with key=> value pairs, that makes up an associative array.

This exercise needs to be completed in multiple steps. You are required to work on every php file in the folder called *YourName*_Week3. Please replace "*YourName*" with your actual name while saving the folder. You will also be required to answer the reflection questions one by one in notes.txt. Please use main.css as your style sheet.

Step 1: Using arrays with key-value pairs to create a table like data structure and displaying this array using HTML table

Please go over the code provided to you in file *index_step1.php.* This file contains the php code to do the following:

Load an array with data

- Create an empty array reference called $employee_salary
- Create new rows in the array by assigning its key=>value pair using a statement similar to *$employee_salary[key] = value*. Assume that the keys are all of string data type. Please note that the keys are unique identifiers for the data set , similar to primary-key of the employee entity. In this example, the unique identifier is the user ID of the employee. Values are the attribute values (in this case, the value of the attribute hourly salary , which is of float data type).

Display the results of the array $employee_salary using an HTML table

  - The table headers (<th> fields) on the first row (<tr> field) are UserId and HourlyPay.
  - The rest of the table rows contain data and are generated using a foreach-loop. The foreach loop splits each row into its key and value pairs. Within the foreach loop, the key is stored in the variable $userId, and value associated with the key in the variable $hourlySalary.
    - Within each row, the table's columns contain the values of userId and hourly salary. The hourly salary is formatted before it is echoed.
    - Please pay attention to the opening and closing of the php tags: <?php ?> the row tags: <tr></tr , the column tags <td></td>, and the table tags

<table></table> . Incorrectly placed tags can result in weird errors, sometimes not displayed by the error statements of the XAMPP.

- Please pay attention to the <?php ?> tags of the foreach loop. You will notice that an addition <?php ?> tag is required where the for loop ends (around the curly brace }).
- Indent your code well – this will make it easy to spot the opening and closing of the tags.

It will help to re-write this code on a separate file and practice writing it yourself.

Things to add to the file - index_step1.php: (2pts)

var_dump the elements of the array by writing the following statements in the php section of the file.

1. var_dump($employee_salary[“jac2233”]);
2. var_dump($employee_salary[“abc4530”]);
3. var_dump($employee_salary[“ghj1238”]);

Then var_dump the whole array as follows**:**

4. var_dump($employee_salary);

Reflection Notes to write in notes.txt

Subtitle: index_step1.php (3 pts)

a. **Write the results of all the four var_dump statements as listed above**
b. **What is the data type of the array’s keys, and values?**

Step 2: Using a 2D array (or, array of arrays) with key-value pairs to create a (multi-column and multi-row)table like data structure and display this array using HTML table

Please open the file *index_step2.php.* This the file contains the code to accomplish the following:

Load an array with data

- Create an empty array referenced by $employee_salary
- Populate $employee_salary with an ‘array of arrays’((2D) array).

- This array emulates a multi-row, multi-column table. This array contains a “super-“key that uniquely identifies each row of the table emulated by the array. Examples of the super-keys / primary-keys are: “jac2233”, “abc 4530”, “ghj1238”
- Each row of the table carries an associative array. This associative array stores data as key->value pairs. Each key of the associative array could be considered as a “sub-“key of the whole 2D array. There are three sub-keys in the example shown in the code and they are “hourlySalary”, “daysPerWeek” and “hoursPerWeek”. These sub-keys could also be called attributes since they represent the attributes of the employee’s salary. In what could be the first row of a table, the ‘super-key’, for example, could be “jac2233”. The sub-keys (or, attributes) are “hourlySalary” , “daysPerWeek” and “hoursPerWeek” and they have the values 56.0, 40, and 5, respectively. In the same sub-key order, rest of the associative array , each emulating the rest of the rows of a table, are maped.

Display the data in array $employee_salary using html.

- Just as in the file, index_step1.php, the array in index_step2.php contains multiple elements/ multiple table rows. Therefore, we will once again need a foreach loop to generate the html rows. This foreach loop splits the $employee_salary array into its “key”=>”value” pairs, that is represented using variables, $userId => $salaryAttributes. The key (,or super-key) uniquelys identify each row. The value, in this case, will be of a different structure than what was there in the foreach loop in index_step1.php. In index_step2.php, the value is not a single value, but a collection of values represented by an associative array. To echo the values of each attribute, we will need to reference each value by its “sub-key”.
- Each attribute value is displayed using echo statements that are embedded within the html tags <td></td>
- The foreach loop is also used to generate the row tags <tr></tr>. There will be as many rows as there are rows in the arrays.
- Please pay attention to the opening and closing of the php tags: <?php ?> the row tags: <tr></tr , the column tags <td></td>, and the table tags <table></table> messing the tags can result in weird errors, sometimes not displayed by the error statements of the XAMPP.
- Please pay attention to the <?php ?> tags of the foreach loop. You will notice that an addition <?php ?> tag is required where the for loop ends (around the curly brace }).
- Indent your code well – this will make it easy to spot the opening and closing of the tags.

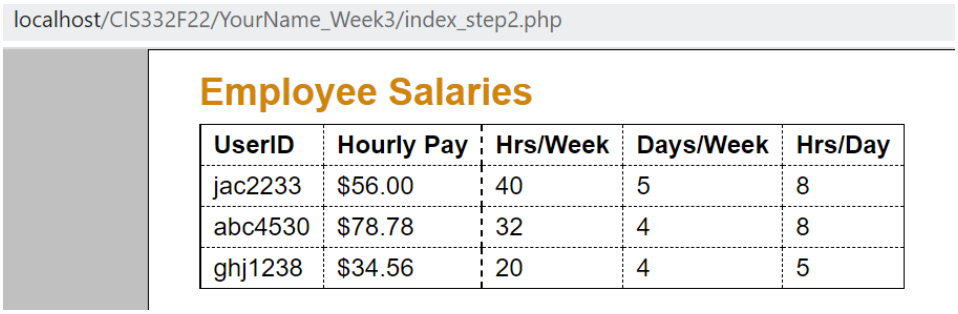
  It will help to re-write this code on a separate file and practice writing it yourself.

To add to the file - index_step2.php:

var_dump the elements of the array by writing the following statements in the php section of the file.

1. var_dump($employee_salary[“jac2233”][“hourlySalary”]); (2pts)
2. In a similar manner, by referencing a data by its super and sub keys, var_dump the values of “daysPerWeek” and “hrsPerWeek” for all the super keys. (6pts)
3. Then var_dump the whole array $employee_salary (2pts)
4. Create another calculated column in the html table and give that column a heading **Hours/Day.**To do this, you will be adding an extra <th></th> tag to the existing ones. (3 pts)
5. Add another row , using the <td></td> tag that does the following using a php script: a)calculate the value of hours per day (by dividing the attributes values) as **hrsPerWeek/daysPerWeek** , b)echo this value with the <td></td> tag. (7 pts)

Your results should looks like this:



Reflection Notes to write in notes.txt

Subtitle: index_step2.php

c. **What will happen if you try to echo $salaryAttributes within another tag <td></td>within the for loop as follows :**

<td><?php echo $salaryAttributes; ?></td>

What kind of error did you get and what does this error mean. Why did you not get a similar error when you tried to echo the $hourlySalary value as : *<td><?php echo $hourlySalary; ?></td>* in index_step1.php ?

Step 3: Allows users to insert a new key=>value pair into the array.

Please open the file index_step3.php. The code in this file will extend what you had written in index_step1.php.

In addition to loading and displaying data as you had done in step 1, you will now include a form to let users input the data (key and value). The data entered into this form will be retrieved (in the same page) using filter_input statements. The key and value will be used to add a new entry into the array.

Create a form to obtain user input

- Create a form that will allow users to enter userId and hourlySalary.
- The form will have the button that allows users to submit the data in the textboxes.
- The form action will go to the same page index_step3.php (for now, we will keep it that way).
- Use “get” method for now – I want you all to get used to watching the url to see the data from the textfields being passed clearly.
- Make sure all the tags are well positioned (especially the <form></form>). If the </form> tage is missing or out of place, data will not be sent properly.

Retrieving user input

- The user input is provided to the server through the fields in the form. These fields include, the text fields – userId, hourlySalary, as well as, the button field called action (and its value “Add Entry”).
- If the user enters data into the text fields and click the button, one should see the field names and the values of the two textfields and the button in the page’s url. Make it a habit to read the url everytime you click a button to send data to the server. The uurl allows to check if the form was submitted with the required field data.
- Use filter_input to get the “action” field value and save it in a variable $action. This variable should have the value of the button, which in this case is “Add Entry”.
- Contingent upon the button click called “Add Entry”, the program retrieves the values of the textboxes userID and hourlySalary. To do this, you have an if-statement that checks if the value of the filtered input variable is “Add Entry”
- Filter the text_fields value from the INPUT_GET. Save the filtered values into the variables $userId and $hourlySalary.
- Add a new entry into the array, as shown in line #27

To add to the file - index_step3.php:

1. var_dump the *$employee_salary* array after adding the new row. It is always a good idea to inspect an array, right after you write any code to change it. (2 pts)
2. You learnt how to validate text field values and give appropriate error statements in week 2 assignment. Use the same methods to validate the filtered inputs. You will write your validation check right after you filter the inputs (inside the *if($action == “Add Entry”) {..}* block .Create a variable called *$error_message* and assign it a default value. Then, use the if ..else if statements to validate the input variables *$userId* and *$hourlyVariable*. If validation fails, populate the *$error_message* string with appropriate error statement. (3 pts)
3. Validation should check the following: (5 pts)
   - userId should be non-empty string
   - hourlySalary should be less than 1000
   - userId should not be a key that already exists in the *$employee_salary* array. To check this, have a foreach loop like what you used to generate the rows in the html code. Make sure to use a different name for key and value (eg you may use *foreach($employee_salary as $id => $value){..}.* Inside the foreach loop, use an if statement to check if *$userId* is equal to the key $id. If the keys match set a boolean

variable/flag (lets, call it) *$isExistId* to true. After the loop runs, check this variable's value and if it is a true, set the *$error_message*. If *$isExistId* remains a false after the loop runs, add the new row, as in line #27. Make sure to initialize the *$isExistID* to false just before the loop.
Note: Instead of using the foreach loop and if condition to check if an array key exists, you could also use the array function called array_key_exists($key, $array) , as explained in the text book chapter on arrays.

4. Write an html code with embedded php script to insert the error message just above the form. (5 pts)

Some of the things you need to pay attention while writing the code are : make sure the field names in the filter_input function are exactly the same as the field names of the respective textboxes. Make sure the button name and value in the form are the same as what you'd use while filtering them in the php code. Once again, I can't stress the importance of paying attention to the tags. The best way to avoid these mistakes is to write the code by yourself, rather than copy-paste them. It is also important to learn how to find out these errors by reading the XAMPP error statements, by watching the url and observing the var_dump results.

Note: While adding the new row into the array, you will notice that the change took place after you submitted the values. However, this change (addition of a new row) does not persist beyond one button-click. This is because arrays do not have the capability of storing values beyond one http transaction. To make the data persistent, we will need additional mechanisms such as sessions, or a database for more powerful persistent data. However, arrays are still very useful. They provide the necessary data structure to store keys and values and to search a given value using the key, in your code. So keep in mind that from here on in this assignment, any changes you make to the array will not persist beyond one button click. The array loads itself afresh every time you run the php code.

Step 4: Add and delete key=>value pairs in an array and update the value using an existing key- all by obtaining the required user inputs.

Please use index_step4.php for this part of the assignment. The code in this file extends the code in index_step3.php.

Allowing user input:

- There are three different forms in the html section of this file to add, delete and update and entry . Each form has its own text boxes and buttons. Please note that the Add Entry form has been modified such that the text box names are different from the ones in index_step3.php. This change in the textbox/field names should be reflected in the filter_input statements that retrieves the data from these text boxes. The Delete form only needs one text box to the get key of the entry to be deleted and the row will be deleted using just the key (using unset function).

- *Note: you will notice that the html code has repetitions. The update and add entry forms are very much alike. While this is not the optimum way to write code, we will use the provided code as a starting point. You could refine your code further by having fewer textboxes and forms.*

- Each form has its button field and all three buttons have the same name “action”. However, the three buttons have different values: “Add Entry” , “Delete Entry” and “Update Entry”.
- Each form has a place for error message . This is done by embedding the $error_message variable using a php script.

Retrieving the user input

- In the given code, lines 20 – 54, implements a filter_input function to get the value of the buttons called action, based on which the further inputs are filtered and validated. Such a code resembles a ‘switch’ (or ‘router’) that contains multiple if-statements that direct (or route) the flow of the code depending on the value of the $action variable. Use of $action variable is common when there are multiple buttons on a page and each button triggers a different set of php code (and even html displays).
- Under each if-statement block, there are distinct filter_input statements corresponding to the distinct textboxes (although , all of them carry key-value data). Once the inputs are filtered , they are saved as variable $userId and $hourlySalary.
- The if-statement block for $action == “Add Entry” will create a new row using the key-value through the statement
  - *$employee_salary[$userId] = $hourlySalary;*
- The if-statement block for $action == “Delete Entry” will create a new row using the key-value through the statements
  - *unset($employee_salary[$userId]);*
  - *$employee_salary = array_values($employee_salary);*
- The if-statement block for $action == “Update Entry” will update an existing row with a given key, by updating to a new value through the statement
  - *$employee_salary[$userId] = $hourlySalary;*

  - *Note: You would have noticed that the update and add entry is done with a similar code. However, update assumes that the key carried by variable $userId already exists in the array , and the add entry will require that the $userId corresponds to a key that is new (or does not already exist in the array). So unless there is good validation to check if the key exists/does not exist, the two code blocks could result in in-correct results.*

To add to the file - index_step3.php:

1. var_dump the *$employee_salary* at the end of each if-statement block. It is always a good idea to inspect an array, right after you write any code to change it. (2 pts)
2. Write your validation checks right after you filter the inputs in each if-statement block. Create a variable called *$error_message* and assign it a default value. Then, use the if ..else if statements to validate the input variables *$userId* and *$hourlyVariable*, for each if-statement block. If validation fails, give populate the *$error_message* string with appropriate error statement. (3 pts)

3. Validation should check the following: (10 pts)
   - userId should be non-empty string
   - hourlySalary should be less than 1000
   - For the “add entry” inputs: userId should not be a key that already exists in the *$employee_salary* array. Please refer to the directions in index_step3.php. If the validation passes, then only add the new row by writing the statement: $employee_salary[$userId] = $hourlySalary;
   - For the “delete entry” input: userId should be a key that already exists in the *$employee_salary* array. Your code will be a minor variation of the directions in index_step3.php. If the validation passes, then only add the new row by writing the statement:
     unset($employee_salary[$userId]);
     $employee_salary = array_values($employee_salary);
   - For the “update entry” inputs: userId should not be a key that already exists in the *$employee_salary* array. Your code will be a minor variation of the directions in index_step3.php. If the validation passes, then only add the new row by writing the statement: $employee_salary[$userId] = $hourlySalary;

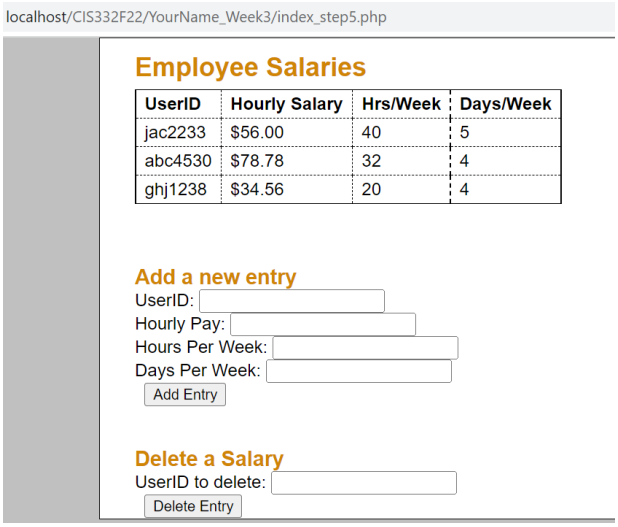
4. Write an html code with embedded php script to insert the error message in the html section of the file (5 pts)

Note: Once you complete the steps 1-5 as mentioned above, if time permits, you may further optimize your code by removing repetitions and redundancies.

Step 5: Add and Delete rows in a using a 2D array.

Please use index_step5.php to complete this part of the assignment. You will be extending the code from index_step2.php. You will use the methods to add and delete entries from steps 3 and 4, by modifying them for a multi-column input of a 2D array that contains super and sub keys.

To add to the file index_step5.php



1. Create the user interface using two forms: One form allows users to enter values for array attributes and the super-key (userID), so that these values can be used to add a new row into the array. The other form allows users to enter the userId (super-key), so that the corresponding row can be deleted. See figure for the fields and button that are required in each form. Just you did in index_step4.php, name all the buttons as “action” . The textboxes may be named differently. Pick textbook names that relate to the data they should contain. (10 pts)

2. Write a statement to retrieve the value of the button (refer to index_step4.php) (2 pts)

3. Write a statement that checks if the value of the action (button) is “Add Entry” and if so, retrieve the values of the text boxes from the corresponding form. Validate these input variables as follows: (5pts)
   - userId: should not be empty.
   - Hourly salary should be numeric and less than 1000
   - hours per week should not exceed 100
   - days per week should not exceed 7
   - userId should not be a super key that already exists.

   If validation fails for any of the cases above, populate an error message string with an appropriate message. Make sure to initialize the error_message string to a default value up in the code.

   If validation passed add a new row using the userId and attribute values. (2 pts)

4. Write a statement that checks if the value of the action (button) is “Delete Entry” and if so, retrieve the values of the text boxes from the corresponding form. Validate these input variables as follows: (5 pts)
   - userId: should not be empty.
   - userId should be a super key that already exists.

   If validation fails for any of the cases above, populate an error message string with an appropriate message. Make sure to initialize the error_message string to a default value up in the code.

If validation passed add a new row using the userId and attribute values.(2 pts)

5. In the html code section embed the error message using php echo statements.(4 pts)

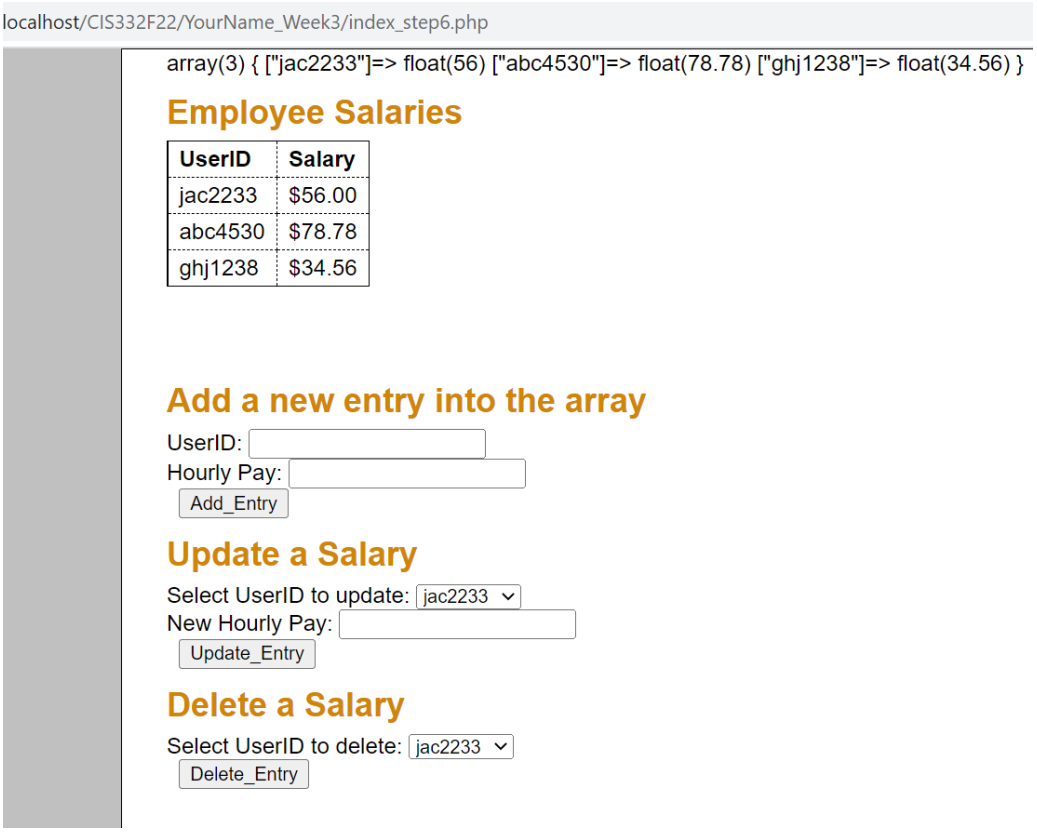
Note: continue to var_dump the variables and the array at various points in the code. This habit could save you from mistakes that you make while altering code, or the ones that could crop up within the html section. This will also teach you to reason the way data is structured and passed on from the forms.

Step 6 : Use a drop-down list in the update and delete forms to get the userId values. Retrieve the values from the drop-down list

Please use the file index_step6.php to complete this exercise. You will modify the php code used to retrieve input data step 4. Rather than retrieve the userId from a text box, you will now get it form a drop-down list.

Include drop-down list in the user interface:

Note: A drop down list is useful to allow user to pick the keys. A dropdown list restricts the keys that the users are forced to pick, so that they do not pass on erroneous values that may not pass validation. In fact, the drop-down list makes validation a bit easier because the code does not need to further check if the userId already exists/ does not exist. In general, a drop-down list just free people from typing code and hence they are user friendly in that respect. Using foreach loop to step through the key-> value pairs of an array and to generate the options of the drop-down list is an important part of many applications.



- Your user interface will look as shown above. The Update Salary and Delete Salary sections now have a drop down list, populated with existing userIds, (as option field of the drop down list). Rest of the fields are the same as in index_step4.php.
- Key things to note about the drop-down list is its name and option values. For example, in lines 100-115, the name of the drop down list is "userId_list1", and option's value is : *"<?php echo $id;?>".* Here $id is the super-key (userId) of the array. The option display is also : *"<?php echo $id;?>"*. This is because we want users to see the userId in the drop down list. If we want users to see a different list, say hourlySalary, then we would code as follows:
    - *<option value="<?php echo $id; ?>">*
    - *<?php echo $salary;?>*
    - *</option>*

  Even in the above case the value is always the super-key/userId
- The dropdown list for the Delete Entry form is also like the one for update, except that the name of the drop-down list is different - *"userId_list2"*.

Retrieve values from the drop-down list

- Just as we did in index_step4.php, retrieve the input values for the three if-statement blocks. The input retrieval, validation and adding a new entry for the case *if($action == "Add Entry"),* is exactly the same as you would do in *index_step4.php.*
- For the if-statement block if*($action == "Update Entry")*, the only change will be : userId is retrieved from the drop down list as follows: *$userId = filter_input(INPUT_GET,*

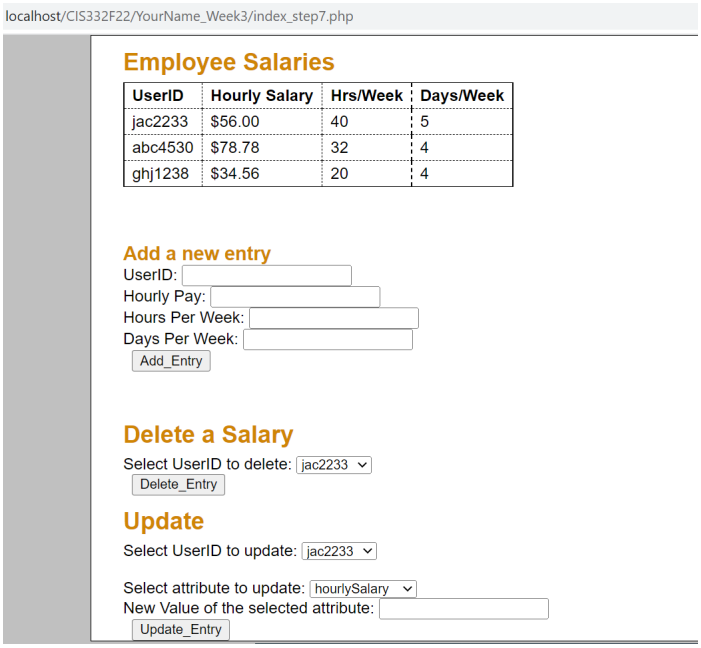
'userId_list1'); . Another change is that while validating, you do not need to check if userId already exists.

- For the if-statement block if*($action == "Delete Entry")*, the only change will be : userId is retrieved from the drop down list as follows: *$userId = filter_input(INPUT_GET, 'userId_list2');* . Another change is that while validating, you do not need to check if userId already exists.

To add to index_step6.php (5 pts)

- Add the php code required to validate the inputs, as you did in index_step4.php. You do not need to validate if an userID exists while updating or deleting a row. Rest of the validations remain the same.
- Provide error _messages when validation fails. Create an error_message string, populate it with appropriate messages and display that to the user when validation fails.
- If validation passes, follow the same procedure as in index_step4.php to either add, or update, or delete an entry.
- Use var_dump for all the variables, including the array. Watch the url after button click and note the fields and their values.

Step 7 : Include dropdown lists in the update and delete forms to get users to pick the userId to update/delete an existing row in a 2D array. Update the row by asking the user the attribute name (sub-key) and its value to update.



To add to index_step7.php

- An Add Entry form with text fields for the userId and rest of the attribute values. This is like what you would have done in index_step5.php (2 pts)
- A Delete Entry form that will now contain a drop-down list for the userId. The code for the drop down list is like what you would have done in index_step6.php. (5 pts)
- An update entry form that will have a drop-down list for the userId. This drop down list is like what you would have done in index_step6.php. (5 pts)
- The update form should have another drop down list with static entries containing the name of the attributes/sub-keys that will need to be modified. To do this, write a simple hard-coded drop-down list (without a foreach loop to generate the options), whose options are : (5 pts)

```
<label>Select attribute to update:</label>
<select name="userId_list4">
        <option
        value="hourlySalary">hourlySalary</option>
        <option
        value="hrsPerWeek">hrsPerWeek</option>
        <option
        value="daysPerWeek">daysPerWeek</option>


</select>
```

- *Note: in the php code I have named my drop-down lists : userId_list2, user_ID_list3 and userID_list4, respectively.*
- Make sure that the three buttons (in the respective forms) have the same name : “action” , but have different values (“Add Entry” , “Delete Entry”, “Update Entry”, respectively). (3 pts)

Retrieve and validate data and perform actions:

- In the php section of the file, you will extend/modify the code from index_step5.php. The if-statement block that adds an entry to the array remains the same , as you would have written in index_step5.php. (3 pts)
- The delete statement block will now retrieve value from the dropdown list , userId_list2 , that is located in the Delete Entry form. You do not need to validate anything, as the userId will not be an empty string or new value anymore. Jump straight to deleting the row using the unset statement. (3 pts)
- The update statement block wasn’t there in index_step5.php. Add an update statement block (as partially written in the code) . The userId and subkey values are retrieved from the drop-down list. You do not need to validate these variables. (3pts)
- Now write an if..else statement that will validate the value in the textbox retrieved from the Update Entry form , depending on what sub-key was selected from the drop down list. (5pts)

    - if($subKey == “hourlySalary”) { // then filter input the textbox and validate as a float. Save this filtered input as a variable $newEntry. }
    - else if ($subKey == “daysPerWeek”) {// then filter input the textbox and validate as an int. Save this filtered input as a variable $newEntry.}
    - else {// then filter input the textbox and validate as an int. Save this filtered input as a variable $newEntry.}
- Then, update the row using the userId and subKey values as: (1 pts)
  //$employee_salary[$userId][$subKey] = $newValue;

Note: To save time, I haven’t included error messages in this step.

Grading :

Step 1: Things to add and reflection notes - 5pts

Step 2: Things to do : 20 pts

Step 3: Things to do: 15 pts

Step 4: Things to do : 20 pts

Step 5 : Things to do : 25 pts

Step 6: Things to do : 5 pts

Step 7: Things to do: 35 pts