IS113 Web Application Development I
Trial Final Exam

Section A: MCQ – Pick ONE Correct Choice (each question worth 2 marks)

1. Given the following directory structure in the web root folder, i.e. (www):

```
+ www
  |- secret
  |    |- papers.php
  |- mystery
  |    |- index.php
```

You are editing the file `papers.php`. Which of the following code snippet(s) will create a hyperlink to `index.php`?

A. `<a href='../index.php'>Mystery</a>`

B. `<a href='/mystery/index.php'>Mystery</a>`

C. `<a href='mystery/index.php'>Mystery</a>`

D. `<a href='/www/mystery/index.php'>Mystery</a>`

E. `<a href='www/mystery/index.php'>Mystery</a>`

2. Given that `index.php` contains the code below:

```
<html><body>
<?php
$arr = [ "apple", "orange", "pear" ];
?>
</body></html>
```

Which of the code snippet(s) when inserted below the above code will display the following output on the browser?

```
apple
orange
pear
```

A.
```
<?php
foreach ( $arr as $item ) {
    echo "$item\n";
}
?>
```

B.
```
<?php
foreach ( $arr as $item ) {
    echo "
          $item";
}
?>
```

C.
```
<?php
for ( $i = 0; $i < count($arr); $i++ ) {
    "$arr[$i]<br>" ;
}
?>
```

D.
```
<?php
$i = 0;
$n = count($arr);
while ($i < $n) {
    echo "$arr[$i++]";
}
?>
```

E. None of the above.

Note: (C) is not an answer because there is no `echo`

3. Given the following directory structure in the web root folder, i.e. (www):

```
+ www
  |- include
  |    |- ConnectionManager.php
  |    |- common.php
  |- src
  |    |- views
  |    |     |- search.php
  |    |- index.php
  |    |- properties.txt
```

You are editing the file `search.php`. Which of the following code snippet(s) will create a reference to `common.php`?

A. `require_once '/common.php';`

B. `require_once '/include/common.php';`

C. `require_once '../include/common.php';`

D. `require_once '.../include/common.php';`

E. `require_once '../../include/common.php';`

4. Given the following code:

```
<?php
// missing

echo "<pre>";
print_r($x);
echo "</pre>";
?>
```

Which of the following code when inserted at the location "// missing" will cause the browser to display the following output:

```
Array
(
    [0] => Array
        (
            [0] => 1
            [1] => 2
            [2] => 3
        )

    [1] => Array
        (
            [0] => 3
            [1] => 4
        )

)
```

A. `$x = [ [1,3], [2,3], [3] ];`

[✓ **B.**] `$x = [ '0'=>[1,2,3], '1'=> [3,4] ];`

C. `$x = [ [0]=>[1,2,3], [1]=>[3,4] ];`

D. `$x = [1,2,3], [3,4];`

E. `$x = [1,2,3,4,5];`

5. Given `index.php`

```
<html>
    <body>
        <a href='middle.php?fruit[]=apple'>x</a>
    </body>
</html>
```

`middle.php`

```
<?php
    header("Location: display.php?fruit[]=orange");
    exit;
?>
```

`display.php`

```
<html><body><ol>
<?php
if (isset($_GET['fruit'])) {
    foreach ($_GET['fruit'] as $item) {
        echo "<li>$item</li>";
    }
} else {
    echo "<li>nothing</li>";
}
?>
</ol></body><html>
```

If the user loads index.php on a web browser and clicks on the x hyperlink, what is the output displayed on the browser?

A.
```
1. apple
2. orange
```

B.
```
1. apple
```

C. [✓]
```
1. orange
```

D.
```
1. nothing
```

E. None of the above

6. Given the following `check.php`

```
<?php

$arr = ["apple" => ''];

if (empty($arr['apple'])) {
    echo "true,";
} else {
    echo "false,";
}

if (in_array('apple', $arr)){
    echo "true,";
} else {
    echo "false,";
}

if (in_array('', $arr)){
    echo "true,";
} else {
    echo "false,";
}

if(array_key_exists('apple', $arr)) {
    echo "true";
} else {
    echo "false";
}

?>
```

What is the output displayed on the browser?

A. `true,true,true,false`

B. `false,false,true,true`

C. `true,true,false,true`

D. `false,true,false,false`

E. None of the above

7. Given the following `form.html`

```
<html><body>
    <form action="process.php" method="post">
        Name <input type='text' name="name" value='      ' /><br/>
  School
      <input type="checkbox" name="school[]" value="sis" />sis
      <input type="checkbox" name="school[]" value="lkcsb" />lkcsb
      <br/>
      <input type="submit" name="SEND" /><br/>
</form>
</body></html>
```

`process.php`

```
<?php
if (isset($_POST['name'])) {
    echo "true,";
} else {
    echo "false,";
}

if (empty($_POST['name'])) {
    echo "true,";
} else {
    echo "false,";
}

if (isset($_POST['school'])) {
    echo "true,";
} else {
    echo "false,";
}

if (empty($_POST['school'])) {
    echo "true,";
} else {
    echo "false,";
}
?>
```

The user loads `form.html` through a web browser, and only just clicks on the submit button.
What will be the output displayed on the browser?

A. `true,true,true,true`
B. `true,false,false,false`
C. `false,false,false,false`
D. `true,true,false,true`
E. None of the above

8. Given the following `form.html`

```
<html>
    <body>
        <form method='post' action='process.php?fruit[]=orange'>
            <input type='text' name='fruit[]' value='apple' />
            <input type='submit'>
        </form>
    </body>
</html>
```

`process.php`

```
<html><body><ol>
<?php
if (isset($_POST['fruit'])) {
    foreach ($_POST['fruit'] as $item) {
        echo "<li>$item</li>";
    }
} else {
    echo "<li>nothing</li>";
}
?>
</ol></body></html>
```

The user loads `form.html` through a web browser, and only just clicks on the submit button. What will be the output displayed on the browser?

A.
```
1. apple
2. orange
```

B.
```
1. apple
```

C.
```
1. orange
```

D.
```
1. nothing
```

E. None of the above

9. Given the following PHP pages:

first.php

```
<?php

session_start();

if (!isset($_SESSION['token'])) {
    $_SESSION['token'] = 'apple';
    header('Location: second.php');
}

$_SESSION['token'] = 'orange';

?>
```

second.php

```
<?php

session_start();

if (isset($_SESSION['token'])) {
    echo $_SESSION['token'];
} else {
    echo 'pear';
}

?>
```

A user accesses the `first.php` page. What will be the output displayed on the browser?

A. apple

[✓ **B.** orange]

C. pear

D. Execution Error

E. None of the above

10. Given the following PHP code :

```
$str1 = '--apple--';
$str2 = '--apple';
$str3 = 'apple--';
$str4 = '--ap--ple--';


if ( /*CONDITION*/ ) {
    echo "apple";
} else {
    echo "orange";
}


function my_trim($value, $ch) {
    $result = '';
    for ($i = 0 ; $i < strlen($value) ; $i++) {
        if ($value[$i] != $ch) {
           $result .= $ch;
        }
    }
    return $result;
}
```

Which code snippets(s) that replaces /*CONDITION*/ in process.php will print `apple`?

i) `trim($str1, '-') == 'apple'`

ii) `trim($str2, '-') == 'apple'`

iii) `trim($str3, '-') == 'apple'`

iv) `trim($str4, '-') == 'apple'`

v) `my_trim($str4, '-') == 'apple'`

A. i

B. i, ii

[✓ **C.** i, ii, iii]

D. v

E. iv, v

Question 1

You are given the following `Product` class.

1. Complete the `hasStock` method. This method returns `true` if there is 1 or more of any size available for purchase. Otherwise, it returns `false`.
2. Complete the `hasStockBySize($pSize)` method. This method returns `true` if stock is available for shirt size `$pSize`. Otherwise, it returns `false`.

testProduct.php

```php
<?php
require_once 'common.php';
$p1 = new Product ( 1, 'SIS T-shirt', 15, ['S'=>1, 'M'=>2, 'L'=>3] );
$p2 = new Product ( 2, 'SMU T-shirt', 17, ['S'=>1, 'L'=>3] );
$p3 = new Product ( 3, 'LKCSB Polo Shirt', 22, [] );

//====================== Testing hasStock() method ======================
// this will return true this shirt has stock available
if( $p1->hasStock() ) {
   echo "{$p1->getName()} is Available";
}
else {
   echo "{$p1->getName()} is NOT available";
}
echo "<br/>";
// this will return false since this shirt has no stock (none of the sizes
are available)
if( $p3->hasStock() ) {
   echo "{$p3->getName()} is Available";
}
else {
   echo "{$p3->getName()} is NOT available";
}
echo '<hr>';
//=============== Testing hasStockBySize($pSize) method ===============
$pSize = 'S';
// this will return true since this shirt has size 'S' stock available
if( $p1->hasStockBySize($pSize) ) {
   echo "Size $pSize of {$p1->getName()} is Available";
}
else {
   echo "Size $pSize of {$p1->getName()} is NOT Available";
}
echo "<br/>";
// this will return false since this shirt does NOT have size 'M' stock
available
$pSize = 'M';
if( $p2->hasStockBySize($pSize) ) {
   echo "Size $pSize of {$p2->getName()} is Available";
}
else {
   echo "Size $pSize of {$p2->getName()} is NOT Available";
}
echo '<hr>';
?>
```

Product.php

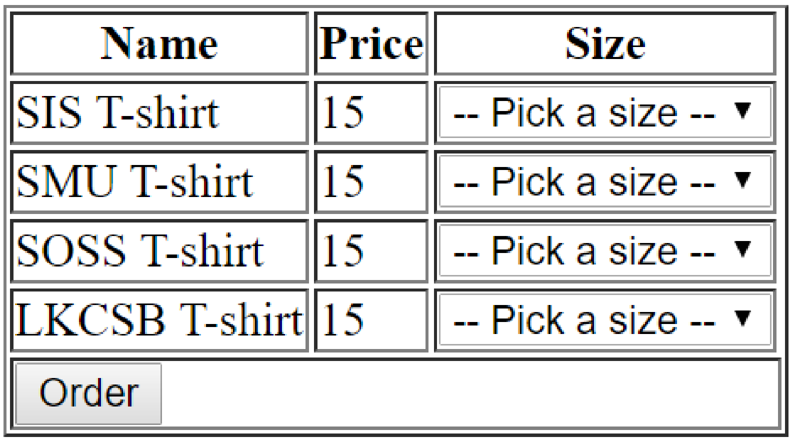
```php
<?php
class Product {
     private $id;    // type: int
    private $name;  // type: str
    private $price; // type: double
    private $stock; // associative array where the
                   // key is the product's size (e.g. 'S', 'M', 'L')
                   // value is the quantity available for purchase(type: int)

    function __construct($id,$name, $price, $stock) {
        $this->id = $id;
        $this->name = $name;
        $this->price = $price;
        $this->stock = $stock;
    }

    public function getID() {
        return $this->id;
    }

    public function getName() {
        return $this->name;
    }

    public function getPrice() {
        return $this->price;
    }

    public function getStock() {
        return $this->stock;
    }

    public function setStock($stock){
        $this->stock = $stock;
    }

    public function hasStock() {
        foreach( $this->stock as $size=>$quantity ) {
            if ($quantity > 0) {
                return true;
            }
        }
        return false;
    }

    public function hasStockBySize($pSize) {
        // Solution 1
        return array_key_exists($pSize, $this->stock)
                 && $this->stock[$pSize] > 0;

        // Solution 2
        return isset( $this->stock[$pSize] )
                 && $this->stock[$pSize] > 0;
    }
}
?>
```

B. Complete the function `print_form($productArray)` in `display.php`. You are given:

```
Class ProductDAO
Methods
   ●  __construct()
      constructs an instance

   ●  reduceInventory($product_id, $size)
      Reduces the inventory count of the product $product_id with the
specified
      $size by 1

   ●  retrieveAll()
      Returns an indexed array of Product objects.
```

1. Takes in one parameter `$productArr` which is an indexed array of `Product` objects.

2. Display the products' details in an HTML table. The dropdown list will contain the list of possible sizes (with available stock) or "`-- Pick a size --`". When the "Order" button is clicked, it sends the product's size (S/M/L) and the product's id over to the order page via HTTP POST request. **Do refer to the code given in part C.** See sample output below:



Name	Price	Size
SIS T-shirt	15	-- Pick a size -- ▼
SMU T-shirt	15	-- Pick a size -- ▼
SOSS T-shirt	15	-- Pick a size -- ▼
LKCSB T-shirt	15	-- Pick a size -- ▼
Order		

```
<?php
// display.php
// code to generate sample output

require_once "common.php";
$dao = new ProductDAO();
$productArr = $dao->retrieveAll();

echo "<html><body>";
print_form($productArr);
echo "</body></html>";

function print_form($productArr) {
  echo "
    <form method='post' action='order.php'>
            <table border='1'>
                <tr>
                    <th>Name</th>
                    <th>Price</th>
                    <th>Size</th>
                </tr>";
  foreach($productArr as $product){
    echo "<tr>
            <td>{$product->getName()}</td>
            <td>{$product->getPrice()}</td>
            <td>
                <select name='item[]'>";
    echo "<option value='' selected disabled>-- Pick a size --</option>";
    foreach($product->getStock() as $size=>$qty) {
       if ($qty > 0) {
          echo "<option
value='{$product->getID()}-{$size}'>$size</option>";
       }
    }
    echo "      </select>
            </td>
        </tr>";
  }
  echo "        <tr>
                    <td colspan='3'>
                        <input type='submit' value='Order'/>
                    </td>
                </tr>
             </table>

        </form>";
}

?>
```

C. Complete the `order.php` page. The page will call the `reduceInventory` method to update the inventory. Dropdown list's value associated with "-- Pick a Size --" should not be processed.

```
<?php

require_once 'common.php';

/*
 * grabs the value send over by the dropdown list
 * the format will be <product's id>-<size>
 * e.g. '1-M'
 */

$items = $_POST['item'];

$dao = new ProductDAO();
foreach($items as $item) {
    if ($item != '') {
        $pieces = explode ("-", $item);
        $dao->reduceInventory($pieces[0], $pieces[1]);
    }
}

header('Location: display.php');
return;
?>
```

Question 2

You are given the following files.

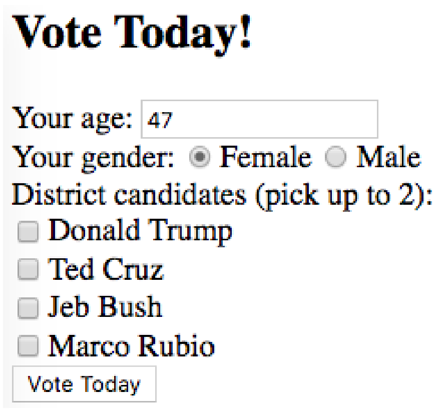
- vote.php
- process_vote.php

vote.php

```
<?php
 session_start();
?>
<html>
<body>
 <h2>Vote Today!</h2>
 <form method='GET' action='process_vote.php'>
   Your age: <input type='text' name='age'><br>
   Your gender: <input type='radio' name='gender' value='Female'>Female
                <input type='radio' name='gender' value='Male'>Male<br>
   District candidates (pick up to 2): <br>
     <input type='checkbox' name='candidates[]' value='Donald Trump'>Donald Trump<br>
     <input type='checkbox' name='candidates[]' value='Ted Cruz'>Ted Cruz<br>
     <input type='checkbox' name='candidates[]' value='Jeb Bush'>Jeb Bush<br>
     <input type='checkbox' name='candidates[]' value='Marco Rubio'>Marco Rubio<br>
   <input type='submit' value='Vote Today'>
 </form>
</body>
</html>
```

In **`vote.php`**, the user must key in **age** and **gender**. The user can select up to **2 candidates**. If the user does not select any candidates, it is a valid input and the server will register it as "The user did not vote for anyone".

A. If the user keys in as shown below, which line(s) in **`process_vote.php`** will break? Circle the line(s) that you believe will break (in the BELOW RECTANGLE).



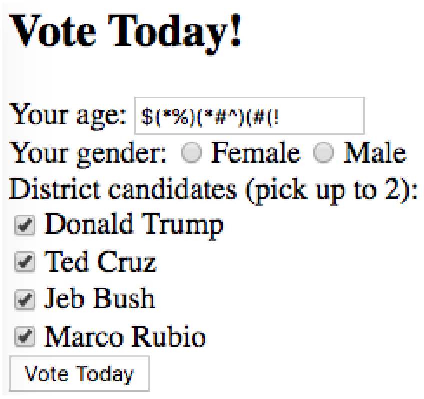
Vote Today!

Your age: 47
Your gender: [✓ Female] Male
District candidates (pick up to 2):
☐ Donald Trump
☐ Ted Cruz
☐ Jeb Bush
☐ Marco Rubio

process_vote.php (CIRCLE BELOW)

```
<?php
session_start();
$errors = [];
$age = $_SESSION['age'];
$gender = $_REQUEST['gender'];
$candidates = $_GET['candidates'];
?>
```

B. If the user keys in as shown below, which line(s) in **`process_vote.php`** will break? Circle the line(s) that you believe will break (in the BELOW RECTANGLE).



Vote Today!

Your age: $(*%)(*#^)(#(!
Your gender: ○ Female ○ Male
District candidates (pick up to 2):
✓ Donald Trump
✓ Ted Cruz
✓ Jeb Bush
✓ Marco Rubio
Vote Today

`process_vote.php (CIRCLE BELOW)`

```
<?php
session_start();
$errors = [];
$age = $_POST['age'];
$gender = $_REQUEST['gender'];
$candidates = $_GET['candidates'];
?>
```

C. Fix both **`vote.php`** and **`process_vote.php`** code so that the following actions are done:

`process_vote.php`

1. **age**: must be integer numeric
2. **voting age**: must be between **18** and **99** (both numbers inclusive).
   a. If the user input is invalid, insert an appropriate error message into **$errors[]**.
3. **gender**: This field is REQUIRED.
   a. If the user input is invalid, insert an appropriate error message into **$errors[]**.
4. The user can select **UP TO TWO (2)** candidates.
   a. If the user does NOT select any candidates, it is a valid input (i.e. the user does not like any of the candidates). If the user input is invalid, insert an appropriate error message into **$errors[]**.
5. The form with any **invalid input values** cannot proceed.
   a. In case of any **invalid input values**, redirect the user back to **`vote.php`**.
6. Note:
   a. Implement **redirect** in such a way that **`vote.php`** can display the **form** with the user's **original input**.
   b. For instance, suppose that the user keyed in the following input values and submitted the form. **`process_vote.php`** should correctly identify **age** to be an invalid input and **candidates** selection to be invalid (the user can choose up to only TWO candidates but this user chose THREE). When the user is redirected back to **`vote.php`**, he/she should be able to see his/her original input reflected in the form.
      i. Age: **a77**
      ii. Gender: **Female**
      iii. Candidates: selected '**Donald Trump**', **'Ted Cruz**' and '**Marco Rubio**'
7. If all form input values evaluate to be **valid**, display "**Thank you for your vote today!**" and display the list of candidates (if any) as an **un-ordered list**.

`vote.php`

1. If this is the first time the user is accessing **`vote.php`**, welcome the user with the greeting message "**Vote Today!**" (heading 2).
2. If the user was redirected to this page due to **invalid input values** that he/she had keyed in previously, do not display the greeting message "Vote Today!". Instead, do the following:
   a. You have **<Number of Errors>** errors in your form. Please rectify them and submit again.
      i. **<Number of Errors>** should be replaced with a numeric value indicating the total number of errors you identified in **`process_vote.php`**.
   b. Next, display the **error messages** as an **ordered list**.
   c. The above 2 points must appear at the top of **`vote.php`**, followed by the user input form at the bottom.
   d. Next, as mentioned above, the user should be able to see his/her original input reflected in the form.

`process_vote.php`

```php
<?php
session_start();
$errors = [];

$age = $_GET['age']; // Maybe consider doing isset()

$gender = '';
$candidate_str = '';
```

```
if (!preg_match("/^[1-9][0-9]*$/", $age) || ($age < 18 || $age > 99)){
    $errors[] = "Age is incorrect";
}
if (!isset($_GET['gender'])){
    $errors[] = "Gender is not specified";
}
else{
    $gender = $_GET['gender'];
}
if(isset($_GET['candidates'])){
    $candidates = $_GET['candidates'];
    if (count($candidates) > 2) {
        $errors[] = "Please enter at most 2 candidates";
    }
    foreach($candidates as $candidate){
        $candidate_str .= "candidates[]=" . $candidate . "&";
    }
}
$_SESSION["errors"] = $errors;

if (count($errors) == 0){
    echo "Thank you for your vote today!<br>";
    echo "<ul>";
    foreach($candidates as $candidate) {
        echo "<li>$candidate</li>";
    }
    echo "</ul>";
}
else{
    header("Location: vote.php?age=$age&{$candidate_str}gender=$gender");
    exit;
}
?>
```

vote.php

```

<?php
 session_start();
?>
<html>
<head>
 <title>Vote Today!</title>
</head>
<body>
 <?php
```

```
    if (!isset($_SESSION["errors"])) {
        echo "<h2>Vote Today!</h2>";
    }
    else{
        $error_count = count($_SESSION['errors']);
        echo "You have $error_count errors in your form. Please rectify them and submit
again.";
        echo "<ol>";
        foreach($_SESSION['errors'] as $error){
            echo "<li>$error</li>";
        }
        echo "</ol>";
        unset($_SESSION["errors"]);
    }

    function printAge() {
        if(isset($_GET["age"])){
            echo "value={$_GET["age"]}";
        }
    }

    function printGender($gender){
        if(isset($_GET["gender"])){
            if ($_GET["gender"] === $gender) {
                echo "checked";
            }
        }
    }

    function printCandidate($candidate){
        if(isset($_GET["candidates"])){
            if (in_array($candidate,$_GET["candidates"])){
                echo "checked";
            }
        }
    }
?>
<form method='GET' action='process_vote.php'>
   Your age: <input type='text' name='age' <?=printAge()?> ><br>
   Your gender: <input type='radio' name='gender' value='Female' <?=printGender('Female')?>
>Female
                <input type='radio' name='gender' value='Male'
<?=printGender('Male')?>>Male<br>
   District candidates (pick up to 2): <br>
```

```
     <input type='checkbox' name='candidates[]' value='Donald Trump' <?=printCandidate('Donald
Trump')?>>Donald Trump<br>
     <input type='checkbox' name='candidates[]' value='Ted Cruz' <?=printCandidate('Ted
Cruz')?>> Ted Cruz<br>
     <input type='checkbox' name='candidates[]' value='Jeb Bush' <?=printCandidate('Jeb
Bush')?>>Jeb Bush<br>
     <input type='checkbox' name='candidates[]' value='Marco Rubio' <?=printCandidate('Marco
Rubio')?>>Marco Rubio<br>
   <input type='submit' value='Vote Today'>
 </form>
</body>
</html>
```

Question 3

Given `extra.php` and `display.php`, suppose that the user attempts to load `display.php` in web browser.

Which of the following code is correct (i.e. the web browser will display "**`Forrest wanna marry Jenny`**"). You can assume `extra.php` and `display.php` are both in the `www` folder.

A.

```php
<?php
// extra.php
session_start();

$_SESSION['forrest_gf'] = 'Jenny';
?>
```

```php
<?php
// display.php
require_once 'extra.php';

echo "Forrest wanna marry {$_SESSION['forrest_gf']}";

?>
```

Answer (Circle one): [✓ Correct] / Incorrect

B.

```php
<?php
// extra.php

$_SESSION['forrest_gf'] = 'Jenny';
?>
```

```php
<?php
// display.php
session_start();
require_once 'extra.php';

echo "Forrest wanna marry {$_SESSION['forrest_gf']}";

?>
```

Answer (Circle one): [✓ Correct] / Incorrect

C.

```php
<?php
// extra.php

echo "Forrest wanna marry {$_SESSION['forrest_gf']}";

?>
```

```php
<?php
// display.php
session_start();
$_SESSION['forrest_gf'] = 'Jenny';
header('Location: extra.php');
?>
```

Answer (Circle one): Correct / [✓ Incorrect]

D.

```php
<?php
// extra.php
session_start();

if (isset($_SESSION['forrest'])) {
    unset($_SESSION['forrest']);
}
if (isset($_SESSION['forrest_gf'])) {
    $_SESSION['forrest_gf'] = 'nobody';
}
echo "Forrest wanna marry {$_SESSION['forrest_gf']}";
?>
```

```php
<?php
// display.php
session_start();
$_SESSION['forrest_gf'] = 'Jenny';
header('Location: extra.php');
?>
```

Answer (Circle one): Correct / [✓ Incorrect]

Question 4

You are given the following file.

- `test.php`

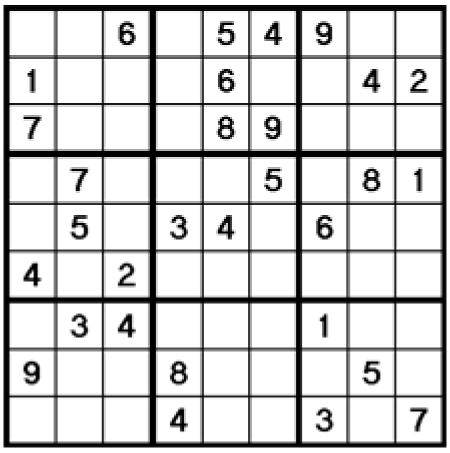
test.php

```
<?php

function spank($weirdos) {
   $weirdos['Eugene'] = 'More like Desmond Tan';

// ADD CODE START
   $weirdos['Eugene'] = str_replace("like", "Like", $weirdos['Eugene']);
   return $weirdos;
// ADD CODE END
}



$weirdos = [ 'Victor' => 'Handsome',
           'Darryl' => 'Not bad',
           'Eugene' => 'Daniel Henney Look',
           'Hong Yang' => 'OMG' ];

$weirdos = spank($weirdos);

echo "{$weirdos['Eugene']}"; // This must echo 'More Like Desmond Tan'
?>
```

In **`test.php`**, add code between "// ADD CODE START" and "// ADD CODE END" so that the last line (**echo…**) echos "**More Like Desmond Tan**".

Question 5

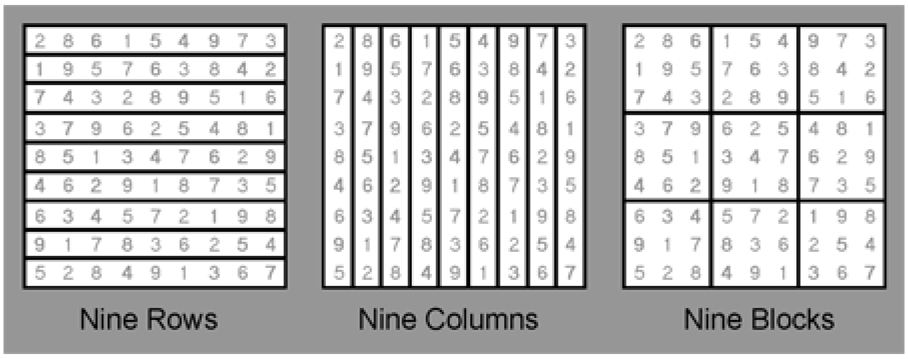
Sudoku is a puzzle game designed for a single player, much like a crossword puzzle. The puzzle itself is nothing more than a grid of little boxes called “cells”. They are stacked nine high and nine wide, making 81 cells total. The puzzle comes with some of the cells (usually less than half of them) already filled in, like this:



		6		5	4	9		
1				6			4	2
7				8	9			
	7				5		8	1
	5		3	4		6		
4		2						
	3	4				1		
9			8				5	
			4			3		7

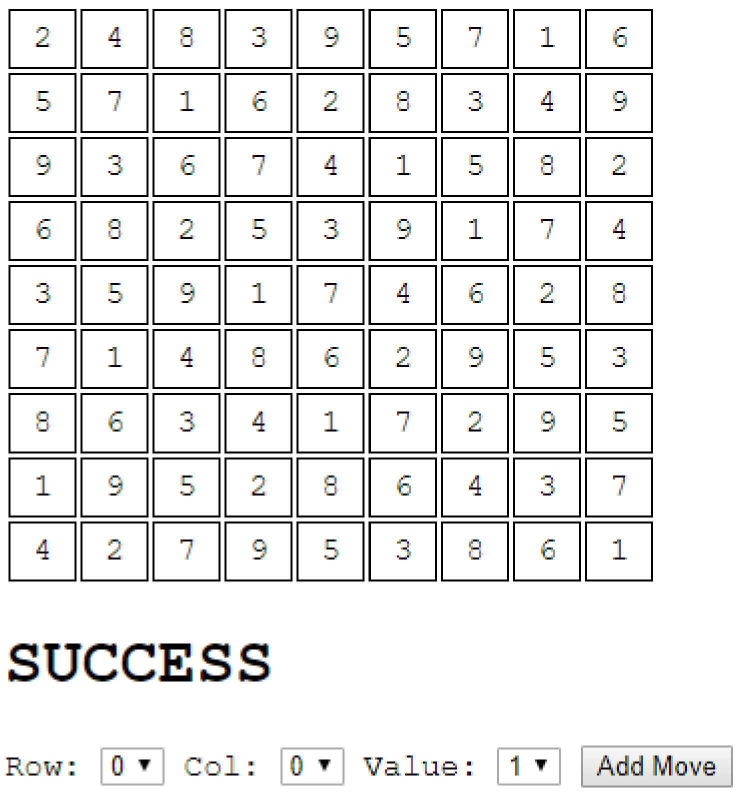
Credit: https://www.learn-sudoku.com/what-is-sudoku.html

For every row, column and block, there should be 1 occurrence of the numbers (1,2,3,4,5,6,7,8,9). No repeat of any numbers is allowed. To put it another way - you must use all nine numbers in each row, column, and block.



Part A: Complete the following methods:

1. `isValid()`
2. `isAllColumnsValid()`
3. `isAllRowsValid()`
4. `isBoxValid()`



2	4	8	3	9	5	7	1	6
5	7	1	6	2	8	3	4	9
9	3	6	7	4	1	5	8	2
6	8	2	5	3	9	1	7	4
3	5	9	1	7	4	6	2	8
7	1	4	8	6	2	9	5	3
8	6	3	4	1	7	2	9	5
1	9	5	2	8	6	4	3	7
4	2	7	9	5	3	8	6	1

SUCCESS

```
<?php
class Sudoku {
    /*
      $board is a 9 x 9 array. An example is as follows:
      [
        ['6', '5', ' ', ' ', ' ', ' ', ' ', ' ', '9'] ,
        ['2', ' ', ' ', '3', ' ', '9', ' ', '6', '5'] ,
        [' ', '9', ' ', '7', ' ', '6', '2', ' ', ' '] ,
        ['7', ' ', ' ', '6', ' ', ' ', ' ', ' ', ' '] ,
        ['3', '8', ' ', '1', ' ', '7', ' ', '9', '4'] ,
        [' ', ' ', ' ', ' ', ' ', '4', ' ', ' ', '7'] ,
        [' ', ' ', '9', '4', ' ', '5', ' ', '8', ' '] ,
        ['8', '7', ' ', '9', ' ', '3', ' ', ' ', '6'] ,
        ['5', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
      ]



      Another example (Insert row0, col2 --> value 8) --> SUCCESS
      [
        ['2', '4', ' ', '3', '9', '5', '7', '1', '6'] ,
        ['5', '7', '1', '6', '2', '8', '3', '4', '9'] ,
        ['9', '3', '6', '7', '4', '1', '5', '8', '2'] ,
        ['6', '8', '2', '5', '3', '9', '1', '7', '4'] ,
        ['3', '5', '9', '1', '7', '4', '6', '2', '8'] ,
        ['7', '1', '4', '8', '6', '2', '9', '5', '3'] ,
        ['8', '6', '3', '4', '1', '7', '2', '9', '5'] ,
        ['1', '9', '5', '2', '8', '6', '4', '3', '7'] ,
        ['4', '2', '7', '9', '5', '3', '8', '6', '1']
      ]
     Note: An empty cell is represented as ' ' (i.e. whitespace)
    */

    private $board;

    public function __construct() {
        $this->board = [
            ['6', '5', ' ', ' ', ' ', ' ', ' ', ' ', '9'] ,
            ['2', ' ', ' ', '3', ' ', '9', ' ', '6', '5'] ,
            [' ', '9', ' ', '7', ' ', '6', '2', ' ', ' '] ,
            ['7', ' ', ' ', '6', ' ', ' ', ' ', ' ', ' '] ,
            ['3', '8', ' ', '1', ' ', '7', ' ', '9', '4'] ,
            [' ', ' ', ' ', ' ', ' ', '4', ' ', ' ', '7'] ,
            [' ', ' ', '9', '4', ' ', '5', ' ', '8', ' '] ,
            ['8', '7', ' ', '9', ' ', '3', ' ', ' ', '6'] ,
            ['5', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '] ,
        ];
    }

    public function getCellValue($row, $col) {
        return $this->board[$row][$col];
    }

    public function setCellValue($row, $col, $value) {
        $this->board[$row][$col] = $value;
    }

    public function getBoardRow($row) {
        return $this->board[$row];
    }
    public function getBoard() {
        return $this->board;
    }

    /**
    * Checks if every row contains the number (1 .. 9) once
    *
    * Return true if every row contains 1 .. 9 once. false otherwise.
    */
    public function isAllRowsValid() {
    // to be completed
        for( $row = 0; $row < 9; $row++ ){ # row number
            $numbers_in_row = [];
            for( $col = 0; $col < 9; $col++ ){ # col number
                $cell = $this->board[$row][$col];
                $numbers_in_row[] = $cell;
            }

            if( !$this->isValidSimple($numbers_in_row) )
                return false;
        }
        return true;
    }

   /*
    * Checks if the sudoku board is valid based on the following
conditions:
    * 1. every row contains the numbers (1 .. 9) once
    * 2. every column contains the numbers (1 .. 9) once
    * 3. every square contains the numbers (1 .. 9) once
    * <p/>
    * Use the following methods to complete it:
    * 1. isAllRowsValid
    * 2. isAllColumnsValid
    * 3. isBoxValid
    *
    * Returns true if the 3 conditions are fulfilled.
```

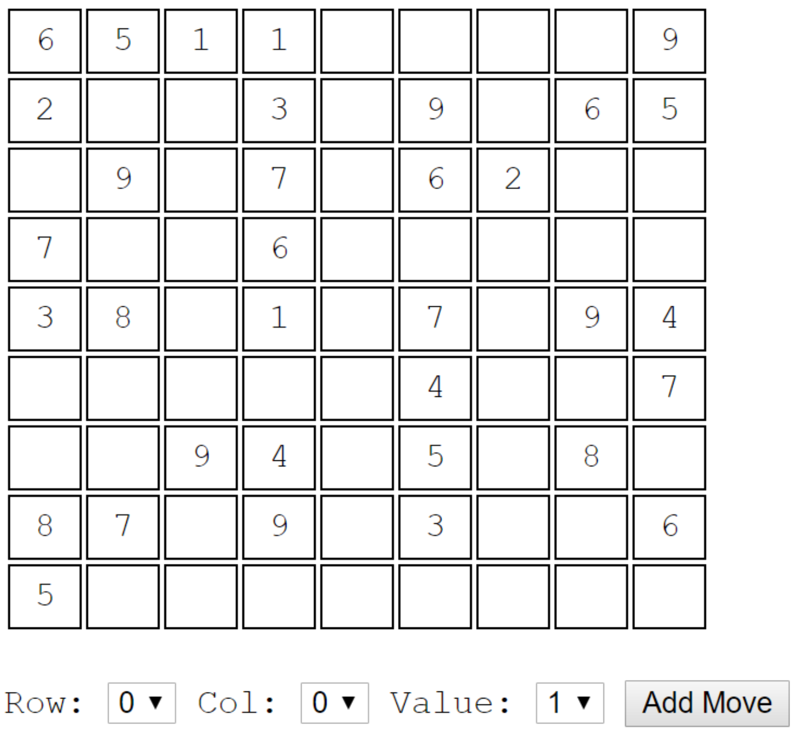
```
  * Otherwise, return false.
  */
  public function isValid() {
     // to be completed
      if( !$this->isAllRowsValid() )
          return false;
      if( !$this->isAllColumnsValid() )
          return false;

      foreach( [0,3,6] as $row ) {
          foreach( [0,3,6] as $col ) {
              if( !$this->isBoxValid($row,$col) )
              return false;
          }
      }
      return true;
 }

 /*
  * Checks if every column contains the number (1 .. 9) once
  *
  * Returns true if every column contains 1 .. 9 once. false otherwise.
  */
  public function isAllColumnsValid() {
      for( $col = 0; $col < 9; $col++ ) { # row number
          $numbers_in_col = [];
          for( $row = 0; $row < 9; $row++ ) { # col number
              $cell = $this->board[$row][$col];
              $numbers_in_col[] = $cell;
          }

          if( !$this->isValidSimple($numbers_in_col) )
              return false;
      }
      return true;
  }

  public function isValidSimple($array){
        foreach( $array as $elem ) {
            if (!is_numeric($elem)) {
                return false;
            }
        }
        for( $i = 1; $i <= 9; $i++ ) {
            if( !in_array($i,$array) )
                return false;
        }
        return true;
    }

   /*
    * Checks if the square contains the number (1 .. 9) once
    * Parameters:
    *
    * $x (type: int) : the row number of the top-left cell of the 3 x 3 box
    * $y (type: int) : the column number of the top-left cell of the
    *                  3 x 3 box
    * Return true if the square contains 1 .. 9 once. false otherwise.
    */
   public function isBoxValid($x, $y) {
        // TO be completed
        $numbers_in_box = [];
        for( $i = 0; $i < 3; $i++ ) {
            for( $j = 0; $j < 3; $j++ ){
                $numbers_in_box[] = $this->board[$x+$i][$y+$j];
            }
        }

        if( !$this->isValidSimple($numbers_in_box) )
            return false;
        else
            return true;
   }

}
```

Part B:

The `SudokuDAO` is written for you with the following methods:

```
Class SudokuDAO
Methods
   ●  __construct()
      constructs an instance

   ●  get()
      Returns a Sudoku object with a sudoku puzzle.
```

You are required to build an PHP page that displays a 9 x 9 sudoku puzzle as an HTML table. The user will select a row value (0-9), column value(0-9) and cell value (1-9) and send it via HTTP POST to itself. It will update the board if the board cell is empty (i.e. the value is a whitespace). If the board meets the criteria (ie. `Sudoku`'s `isValid` method), it updates `$message` with the **"SUCCESS"** string, and clears the `Sudoku` object from the Session. You are **NOT** required to handle the situation whereby all the values are entered, and the Sudoku board is invalid. The user will just have to restart his browser.



6	5	1	1					9
2			3		9		6	5
	9		7		6	2		
7			6					
3	8		1		7		9	4
					4			7
		9	4		5		8	
8	7		9		3			6
5								

Row: 0 ▾ Col: 0 ▾ Value: 1 ▾

```
<?php
require_once 'Sudoku.php';
require_once 'SudokuDAO.php';

session_start();

if (!isset($_SESSION['current-game'])) {
    $dao = new SudokuDAO();
    $sudoku = $dao->get();
    $_SESSION['current-game'] = $sudoku;
}

$sudoku = $_SESSION['current-game'];

/*
  Task:
  Gets the row, col and cell value from the $_POST object(if they are sent
  over). It will update the board if the board cell is empty
  (i.e. whitespace).
*/

if( isset($_POST['row']) ) {
    $row = $_POST['row'];
    $col = $_POST['col'];
    $value = $_POST['value'];

    if ( $sudoku->getCellValue($row, $col) == ' ' ) {
        $sudoku->setCellValue($row, $col, $value);
    }
}

$message = "";
if ( $sudoku->isValid() ) {
    $message = "SUCCESS";
    unset($_SESSION['current-game']);
}


/*
  This will print the 9 x 9 suduko as an HTML table
 */
function generate_board($sudoku) {
    echo "<table>";
    for( $row = 0; $row < count($sudoku->getBoard()); $row++ ) {
        echo "<tr>";
        for( $col = 0; $col < count($sudoku->getBoardRow($row)); $col++ ) {
            echo "<td>{$sudoku->getCellValue($row, $col)}</td>";
        }
        echo "</tr>";
    }
    echo "</table>";
}

?>
<html>
<head>
    <style>
     td {
            border: 1px black solid;
            width: 30px;
            height: 30px;
            text-align: center;
     }
     body {
            font-family: courier;
     }
    </style>
</head>
<body>

<?php
    generate_board($sudoku);
    echo "<h1>$message</h1>";
?>

<form method='post'>
    Row: <select name='row'>
        <option value='0'>0</option>
        <option value='1'>1</option>
        <option value='2'>2</option>
        <option value='3'>3</option>
        <option value='4'>4</option>
        <option value='5'>5</option>
        <option value='6'>6</option>
        <option value='7'>7</option>
        <option value='8'>8</option>
    </select>
    Col: <select name='col'>
        <option value='0'>0</option>
        <option value='1'>1</option>
        <option value='2'>2</option>
        <option value='3'>3</option>
        <option value='4'>4</option>
        <option value='5'>5</option>
        <option value='6'>6</option>
        <option value='7'>7</option>
        <option value='8'>8</option>
    </select>
    Value: <select name='value'>
        <option value='1'>1</option>
        <option value='2'>2</option>
        <option value='3'>3</option>
        <option value='4'>4</option>
        <option value='5'>5</option>
        <option value='6'>6</option>
        <option value='7'>7</option>
        <option value='8'>8</option>
        <option value='9'>9</option>
    </select>
    <input type='submit' name='action' value='Add Move' />
</form>
</body>
</html>
```

Question 6

Given the following:

a. Class **`User`** `(in User.php)`
   Attributes
   - `username`
     - Username of a staff
   - `passwordHash`
     - Hash of password using PHP function `password_hash()`.
   - `employeeType`
     - Has value `'normal'`, `'management'` or `'inactive'`

   Methods
   - `__construct($username,$passwordHash,$employeeType)`
     - Constructs an instance
   - `getUsername()`
     - Returns a `User` object's `$username` property.
   - `getPasswordHash()`
     - Returns a `User` object's `$passwordHash` property.
   - `getEmployeeType()`
     - Returns a `User` object's `$employeeType` property.

b. Class **`UserDAO`** `(in UserDAO.php)`
   Methods
   - `get($username)`
     - Returns a `User` object for the user with username specified by parameter `$username`.
     - If there is no such username, returns `false`.

c. `create.sql` - creates database table.

d. `common.php` - refer to the appendix.

e. `index.php` is the login form and submits to `login.php`. Details below.

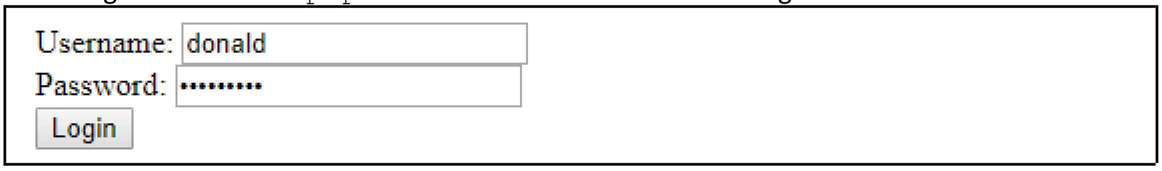
f. `login.php` authenticates the user and redirects to the main page. Details below.

g. Contents of `protect.php` is shown below.
   a. It will be included in all pages that require protection against unauthenticated access.
   b. It works the same way regardless of employee type.

```
<?php
require_once "common.php";

if ( !isset($_SESSION['login']) ) {
    header('Location: index.php');
    exit();
}
?>
```

Login flow

1. User logins via `index.php` which has a form with the following fields:



    a. Text box for username.

    b. Text box for password and the entered value must be masked (display asterisks).

    c. Submit button that submits the form to `login.php` via HTTP POST.

2. The file `login.php` retrieves the values from the submitted form, and authenticates the username and password by retrieving the `User` object for the username using `UserDAO`.

3. If login is successful,

    a. If the user's `employeeType` is `'management'` or `'normal'`,
        i. Add the necessary session variable and assign the `User` object to it such that `protect.php` (as given above) will work as intended.

        ii. If the user's `employeeType` is `'management'`, go to `management_main.php`

        iii. If the user's `employeeType` is `'normal'`, go to `normal_main.php`

    b. Otherwise,
        i. Redirects user back to `index.php` . The URL displayed on the browser address bar is `http://localhost/index.php`.

        ii. The page `index.php` displays an error message "`Access denied`" below the form.

4. If login fails,
    a. Redirects user back to `index.php` . The URL displayed on the browser address bar is `http://localhost/index.php`.

    b. The page `index.php` displays an error message "`Login fails`" below the form.

Instructions
You are to update `index.php` and `login.php` below such that the 2 files work as described above.

1. Make the minimal amount of code changes and additions to make the two PHP pages work as described in the above flow.

2. If there are any errors in the existing code, modify (annotate on top of the existing code) the code to make it work. E.g.
~~display "I love "PHP"!";~~ echo 'I love "PHP"!';

3. Marks will be deducted if you change correct code into incorrect code, if you change any working code unnecessarily, or if you add unnecessary code.

4. You MUST fill in the respective codes in the specified section.
E.g. `// get submitted form values` – You must place code to get form values in that section. Marks will not be awarded for placing it elsewhere.

// highlighted parts are errors to be corrected. Answers are in blue
// marks are not shown for the 2 pages. Otherwise, the number of errors can be guessed

index.php

```
<?php

require_once "common.php";

?>

<html>

<body>

    <form action="login.php">
    //  <form action="login.php" method="post">

        Username <input type="text" name="username" /><br/>

        Password <input type="mask" name="password" /><br/>
        // Password <input type="password" name="password" /><br/>

        <input type="submit" value="Login" />

    </form>

    <?php

        printErrors();

    ?>

</body>

</html>
```

login.php

```
<?php

//   require_once "common.php";

// get submitted form values

$username = $_GET['username'];
// $username = $_POST['username'];

$password = $_REQUEST['password'];


// get User object

$userDAO = new UserDAO();
```

```
$user = $userDAO->get( $userName);
//  $user = $userDAO->get( $username);


// verify username and password

$errors = [];

if (
     $user !== false

     ||  password_verify( $password, $user->getPasswordHash() )
     //  && password_verify( $password, $user->getPasswordHash())

) {

    // based on employee's type, identify the main page to go to.

    if ( $user->getEmployeeType() == 'management') {

        $url = 'management_main.php';

    }

    if ( $user->getEmployeeType() == 'normal') {
    //  elseif ( $user->getEmployeeType() == 'normal') {


        $url = 'normal_main.php';

    }

    else {

        $errors[] = 'Access denied';

    }

} else {

    $errors[] = 'Login fails';

}


// if there are errors

if ( empty( $errors ) ) {

    $_SESSION['user'] = $user;
    //  $_SESSION['login'] = $user;

} else {

    $_SESSION['errors'] = $errors;
    $url = 'index.php';

}


// redirect to target page

location( $url );
//  header( "Location: $url" );
exit;
?>
```

Appendix A. PHP quick reference

Operators precedence

The following table lists the operators in order of precedence, with the highest-precedence ones at the top. Operators on the same line have equal precedence, in which case associativity decides grouping.

Associativity	Operators	Additional Information
non-associative	new	new
Left	[	array()
Right	**	arithmetic
Right	++ -- ~ (int) (float) (string) (array) (object) (bool) @	types and increment/decrement
non-associative	instanceof	types
Right	!	logical
Left	* / %	arithmetic
Left	+ - .	arithmetic and string
Left	<< >>	bitwise
non-associative	< <= > >=	comparison
non-associative	== != === !== <> <=>	comparison
Left	&	bitwise and references
Left	^	bitwise
Left	|	bitwise
Left	&&	logical
Left	||	logical
Right	??	comparison
Left	? :	ternary
Right	= += -= *= **= /= .= %= &= |= ^= <<= >>=	assignment
Left	and	logical
Left	xor	logical
Left	or	logical

array_key_exists

```
bool array_key_exists ( mixed $key , array $array )
```

array_key_exists() returns TRUE if the given key is set in the array. key can be any value possible for an array index.

array_slice

```
array array_slice ( array $array , int $offset [, int $length = NULL [, bool
$preserve_keys = FALSE ]] )
```

array_slice() returns the sequence of elements from the array `array` as specified by the `offset` and `length` parameters.

count

```
int count ( mixed $array_or_countable [, int $mode = COUNT_NORMAL ] )
```

Counts all elements in an array, or something in an object.

empty

```
bool empty ( mixed $var )
```

Determine whether a variable is considered to be empty. A variable is considered empty if it does not exist or if its value equals FALSE. empty() does not generate a warning if the variable does not exist.

explode

```
array explode ( string $delimiter , string $string )
```

Returns an array of strings, each of which is a substring of string formed by splitting it on boundaries formed by the string delimiter.

implode

```
string implode ( string $glue , array $pieces )
string implode ( array $pieces )
```

Join array elements with a glue string.
glue defaults to an empty string.

in_array

```
bool in_array ( mixed $needle , array $haystack [, bool $strict = FALSE ] )
```

Searches haystack for needle using loose comparison unless strict is set.

isset

```
bool isset ( mixed $var [, mixed $... ] )
```

Determine if a variable is set and is not NULL.

If multiple parameters are supplied then isset() will return TRUE only if all of the parameters are set. Evaluation goes from left to right and stops as soon as an unset variable is encountered.

Returns TRUE if var exists and has value other than NULL. FALSE otherwise.

password_hash

```
string password_hash ( string $password , integer $algo [, array $options ] )
```

Returns a string containing the hash of $password.
E.g. `password_hash( "password", PASSWORD_BCRYPT )`

password_verify

```
boolean password_verify ( string $password , string $hash )
```

Verifies that the given hash matches the given password.
Returns TRUE if the password and hash match, or FALSE otherwise.

sizeof
```
int sizeof ( mixed $array_or_countable [, int $mode = COUNT_NORMAL ] )
```

Counts all elements in an array, or something in an object.

str_ireplace
```
mixed str_ireplace ( mixed $search , mixed $replace , mixed $subject [, int
&$count ] )
```

This function returns a string or an array with all occurrences of search in subject (ignoring case) replaced with the given replace value.

Returns a string or an array of replacements.

str_replace
```
mixed str_replace ( mixed $search , mixed $replace , mixed $subject [, int
&$count ] )
```

This function returns a string or an array with all occurrences of search in subject replaced with the given replace value.

This function returns a string or an array with the replaced values.

stripos
```
mixed stripos ( string $haystack , string $needle [, int $offset = 0 ] )
```

Find the numeric position of the first occurrence of needle in the haystack string.

Unlike the strpos(), stripos() is case-insensitive.

Returns the position of where the needle exists relative to the beginning of the haystack string (independent of offset). Also note that string positions start at 0, and not 1.

Returns FALSE if the needle was not found.

strlen
```
int strlen ( string $string )
```

Returns the length of the given string.

strpos
```
mixed strpos ( string $haystack , mixed $needle [, int $offset = 0 ] )
```

Find the numeric position of the first occurrence of needle in the haystack string.

Returns the position of where the needle exists relative to the beginning of the haystack string (independent of offset). Also note that string positions start at 0, and not 1.

Returns FALSE if the needle was not found.

strtolower

```
string strtolower ( string $string )
```

Returns string with all alphabetic characters converted to lowercase.

strtoupper

```
string strtoupper ( string $string )
```

Returns string with all alphabetic characters converted to uppercase.

substr

```
string substr ( string $string , int $start [, int $length ] )
```

Returns the portion of string specified by the start and length parameters.

trim

```
string trim ( string $str [, string $character_mask = " \t\n\r\0\x0B" ] )
```

This function returns a string with whitespace stripped from the beginning and end of str. Without the second parameter, trim() will strip these characters:

- " " (ASCII 32 (0x20)), an ordinary space.
- "\t" (ASCII 9 (0x09)), a tab.
- "\n" (ASCII 10 (0x0A)), a new line (line feed).
- "\r" (ASCII 13 (0x0D)), a carriage return.
- "\0" (ASCII 0 (0x00)), the NUL-byte.
- "\x0B" (ASCII 11 (0x0B)), a vertical tab.

Appendix B. HTML cheat sheet

Modified from http://www.simplehtmlguide.com/cheatsheet.php

Document outline

<html> ... </html>	HTML document
<head> ... </head>	Page information
<body> ... </body>	Page contents

Section Divisions

<div> ... </div>	Division or Section of Page Content
<span> ... </span>	Section of text within other content
<p> ... </p>	Paragraph of Text
<br>	Line Break
<hr>	Basic Horizontal Line

Links

<a href="url"> link name </a>	Create a link to another page or website
<a name=”anchor”>	Anchor
<a href=”#anchor”>	Link to anchor

Text Formatting

<h?> ... </h?>	Heading (?= 1 for largest to 6 for smallest, eg h1)
<b> ... </b>	Bold Text
<i> ... </i>	Italic Text
<u> ... </u>	Underline Text
<strike> ... </strike>	Strikeout
<sup> ... </sup>	Superscript - Smaller text placed below normal text
<sub> ... </sub>	Subscript - Smaller text placed below normal text
<small> ... </small>	Small - Fineprint size text
<pre> ... </pre>	Pre-formatted Text
<strong> ... </strong>	Strong - Shown as Bold in most browsers
<em> ... </em>	Emphasis - Shown as Italics in most browsers

Lists

<ol> ... </ol>	Ordered List
<ul> ... </ul>	Un-ordered List
<li> ... </li>	List Item (within ordered or unordered)

Forms

<form> ... </form>	Form input group declaration
<input> ... </input>	Input field within form
<label> ... </label>	Input label
<select> ... </select>	Select options from drop down list
<option> ... </option>	Option (item) within drop down list
<textarea> ... </textarea>	Large area for text input

Tables

<table> ... </table>	Define a Table
<tr> ... </tr>	Table Row within table
<th> ... </th>	Header Cell within table row
<td> ... </td>	Table Cell within table row

Image

<img src="filename.jpg">	Show an image

Appendix C. common.php

Unless specified otherwise, you may assume the file `common.php` is available and to be used where relevant. Its contents are as shown below.

```
<?php
date_default_timezone_set('Asia/Singapore');

spl_autoload_register(
	function($class) {
		$path = $class . ".php";
		require_once $path;
	}
);

session_start();

function printErrors() {
	if(isset($_SESSION['errors'])){

	echo "<ul style='color:red;'>";

	foreach ($_SESSION['errors'] as $error) {
		echo "<li>" . $error . "</li>";
	}

	echo "</ul>";
	unset($_SESSION['errors']);
	}
}
?>
```

THE END