# Everlytic Developer Assessment

|  |  |
| --- | --- |
| Name | Omphemetse Samuel Lawrence Mafoko |
| Date | 17 August 2020 |

Please complete the answers to the questions below. The assessment should take roughly 30 minutes.

### What is the difference between public, protected and private in a class definition?

|  |
| --- |
| Public definition means that the variable or method being defined with this access type can be accessible both from inside and outside the class. They can be accessed the class itself as a parent class and its children can also access them too. They can also be accessed by other classed that do not fall under the scope of this main classes children. Being public gives it the capabilities to be accessible system wide. |
| Protected definition means that the variable or method being defined with this access type can be accessible both from inside and outside the class by only the main class itself and its children. Being protected does give it the capabilities to be accessible system wide, only the class itself and the children that fall under its scope will have access to these properties.. |
| Private definition means that the variable or method being defined with this access type can only be accessible from within the class itself only. No children falling under its scope will have access to these proper ties and the properties are not accessible system wide. |

### Given this code: function doSomething(&$foo) { $bar = $foo; $foo += 1; return $foo; } $value = 3; $result = doSomething($value); echo "value: $value, result: $result"; What will be output to screen and why?

|  |
| --- |
| The output of the screen will be 4 for both value and result variables. Pass by reference or pass by address pointer is used here and this is how it is implemented. The address passed here give capabilities to change the value stored at that location directly instead copying it to a different location of a new variable and change it there. Therefore one memory place is referenced and the value inside that memory located can be altered directly without being copied. This is how it happens in C language. |
|  |
|  |

### What is wrong with this query: "SELECT \* FROM table WHERE id = $\_POST[ 'id' ]"?

|  |
| --- |
| $\_POST[‘id’] is used directly within an SQL query without the post property being cleaned or sanitized first to eliminate SQL injection on the payload that can be send direct to the database and can cause a lot of damage to the database or which can compromise the database.  An attacker can easily take advantage of this table and database due to its lack of security. The best approach will be to always clean, filter out or sanitize all payload input coming from the client. Th general rule of thumb is that never trust user or client input even if you know them as a trust-able source. Always clean all input payloads that you receive to avoid any form of attack that can be launched to your system and one of those attacks is SQL injection which can be used in this case. |
|  |
|  |

### What is the cause of this warning: 'Warning: Cannot modify header information - headers already sent', and what is a good practice to prevent it?

|  |
| --- |
| $bar is not declared as static that's whats wrong. Any method declared as static is accessible without the creation of an object. Static function are associated with the class, not an instance of the class. They are permitted to access static methods and static variables. To add a static method to the class, static keyword is used. Therefore $bar should be declared as static. |
|  |
|  |

### What is wrong with this code: class Foo { protected $bar; public function \_\_construct() { $this->bar = 1; } public static function doSomething() { return $this->bar; } }

|  |
| --- |
| $bar is not declared as static, that's whats wrong here. To fix this issue declare bar as static. |
| Any method declared as static is accessible without the creation of an object. Static functions are associated with the class, not an instance of the class. They are permitted to access only static methods and static variables. To add a static method to the class, static keyword is used. |
|  |

### Write a program that prints the numbers from 1 to 100. But for multiples of three print "Fizz" instead of the number and for the multiples of five print "Buzz". For numbers which are multiples of both three and five print "FizzBuzz".

<?php

for ($x = 1; $x < 101; $x++) {

for ($y = 1; $y < 11; $y++) {

if (($x == 3 || $x == 5) && $x == $y) {

if ($x == 3) {

$value = "Fizz";

} else {

$value = "Buzz";

}

} else {

if ($y == 3 || $y == 5) {

$value = "FizzBuzz";

} else {

$value = $x \* $y;

}

}

echo $x.' x '.$y.' = ' . $value, PHP\_EOL;

//$results =

// (($x == 3 || $x == 5) && $x == $y) ? ($x == 3) ? "Fizz" : "Buzz" :

// (($y == 3 || $y == 5)) ? "FizzBuzz" : $x \* $y;

//echo $x.' x '.$y.' = ' . $results, PHP\_EOL;

}

echo PHP\_EOL;

}

### **What does the following code do? Explain what’s going on there.**

$date = '08/26/2003';

print preg\_replace('/([0-9]+)\/([0-9]+)\/([0-9]+)/'‚ '$2/$1/$3', $date);

|  |
| --- |
| The following code receives date in the format 08/26/2003 the give the output of the same date in this format 26082003. The regular expression only allows or check for numeric numbers separated by a forward slash and if the value before or after the forward slash is a number, it then used substitution in this case $2$1$3 to convert the number to 26082003.  With the substitutions, expression number 2 is extracted using $2, expression number 1 is extracted using $1 and expression number 3 is extracted using $3 and that ends up converting the date from 08/26/2003 to 26082003.   1. Expression : ([0 – 9]+)\/([0 – 9]+)\/([0 - 9]+) 2. Substitution : $2$1$3 3. String : 08/26/2003 4. Results : 26082003 |
|  |

### **Given a line of text $string, how would you write a regular expression to strip all the HTML tags from it?**

|  |
| --- |
| $str = “Hello <b>world</b> program;  echo htmlspecialchars($str); |
| Will results to : Hello &lt;b&gt;world&lt;/b&gt; program |
|  |

### A palindrome is a word that reads the same backward or forward. Write a function that checks is a given word is a palindrome. Characters case should be ignored. EG. Deleveled is a palindrome and should return true as character case is ignored.

### <?php

class Palindrome

{

public static function isPalindrome($word)

{

echo self::checkPalindrome($word), PHP\_EOL;

}

public function checkPalindrome($word) {

$str\_len = strlen($word);

echo PHP\_EOL, PHP\_EOL , "=================================================", PHP\_EOL;

echo "Received word evaluated : ". $word, PHP\_EOL;

if ($str\_len % 2 == 0) {

$middle\_char\_pos = floor($str\_len / 2);

echo "World length is even, middle character at position : $middle\_char\_pos", PHP\_EOL;

} else {

$middle\_char\_pos = ceil($str\_len / 2);

echo "World length is odd, middle character at position : $middle\_char\_pos", PHP\_EOL;

}

$left\_str = $str\_len % 2 == 0 ? substr($word, 0, $middle\_char\_pos - 0) : substr($word, 0, $middle\_char\_pos - 1);

$right\_str = substr($word, $middle\_char\_pos, strlen($left\_str));

$right\_str\_cnt = strlen($right\_str);

echo "-------------------------------------------------", PHP\_EOL;

echo "Left String Original : ". $left\_str, PHP\_EOL;

echo "Right String Original : ". $right\_str, PHP\_EOL;

echo "-------------------------------------------------", PHP\_EOL;

echo "Left String To Be Used To Indicate Comparison - Original : ". $left\_str, PHP\_EOL;

$left\_str = strtolower($left\_str);

$right\_str\_reversed = strtolower(strrev($right\_str));

echo "Right String To Be Used To Indicate Comparison - Reversed : ". $right\_str\_reversed, PHP\_EOL;

echo "-------------------------------------------------", PHP\_EOL;

$left\_str\_array = str\_split($left\_str);

$array\_count = count($left\_str\_array);

$array\_count = $array\_count - 1;

$match\_count = 0;

$start\_pos = 0;

foreach($left\_str\_array as $x => $left\_char) {

if ($x == 0) {

$right\_char = substr($right\_str, -1);

} else {

$right\_char = substr($right\_str, -($start\_pos + 1), -$start\_pos);

}

$start\_pos++;

$right\_str\_cnt--;

echo "$x. $left\_char = $right\_char", PHP\_EOL;

if($left\_char == $right\_char) {

$match\_count++;

}

}

echo "-------------------------------------------------", PHP\_EOL;

if ($match\_count === ($array\_count + 1)) {

return $word." - It's a palindrome.";

} else {

return $word." - Sorry, that's not a palindrome.";

}

}

}

echo Palindrome::isPalindrome('111111');

echo Palindrome::isPalindrome('Deleveled');

echo Palindrome::isPalindrome('Red rum, sir, is murder');

echo Palindrome::isPalindrome('mom');

echo Palindrome::isPalindrome('mommom');

echo Palindrome::isPalindrome('313');

echo Palindrome::isPalindrome('Eva, can I see bees in a cave?');

echo Palindrome::isPalindrome('Omphemetse');

### What security issue is prevalent in the code below and how would you fix it? <?php $messageStmt = $db->query('select message\_text from messages where message\_id = 1'); $messageStmt->execute(); $message = $messageStmt->fetch(PDO::FETCH\_OBJ); ?> <div><?= $message->message\_text; ?></div>

### The statement could lead to SQL Injection if input payload is just used directly on the statement. The best solution for this issue will be to use prepared statements where input will be prepared separately and guarded against any form of anomality then after the input is prepared and sanitized, then in can be used on the SQL query with PHP. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Write an inner join for the following tables

|  |
| --- |
| SELECT usr.UserKey, usr.FirstName, usr.LastName, usr.Email, adr.Address1, adr.Address2, adr.City, adr.State, adr.ZipCode  FROM User usr  INNER JOIN Address adr ON usr.UserKey = adr.UsrKey; |
|  |

### 