LAB #4 – PHP (console)

This task needs to be completed using a text-redactor or IDE (e.g., Visual Code). Before you start your lab task, please **fully read** this specification! You are strongly advised to refer to lecture materials. The tasks specified here should be run with **PHP on command line**.

Before you start

Establish a new folder for your lab, where you place all the files connected to this task.

Ensure you have a SSH client to access *enos.itcollege.ee*, or locally running PHP installation on your machine. If you do not have a SSH client, you can download Putty (an SSH and telnet client) from https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html. The *exe* can be run without installation!

Connect to *enos.itcollege.ee* and test that PHP is available for you. You can do this by typing "`php –v`" on the command line.

Task 1 – Implement Program 1: Arrays, Loops, Conditions

Create a new PHP file in your lab folder. Implement a program to carry out the following tasks:

1. Set error reporting to ALL.
2. Declare a constant stating the conversion coefficient to convert kilometres into miles. The value of the constant is 1.60934.
3. Create variable `$numOfDistances` carrying a pseudo-random value between 5 and 20;
4. Create an array called `$arrDistances`, and fill it with pseudo-random integer numbers between 1 and 100. The size of the array is determined by the running value of `$numOfDistances`. Use the `array_push()` function to fill the array.
5. Output the array to console, e.g. using the `print_r()` function.
6. Sort the array elements into ascending order by their values, and output the sorted array. There is a special function for sorting in PHP, see the manual.
7. Now create a new array `$arrMiles`. In this array the key of an array element is the value in kilometres from `$arrDistances`, and the value corresponding to the key is miles (kilometres converted into miles). Use the declared constant (p.2) for this task. Write a loop for this task. You can acquire the size of the `$arrDistances` using the `count()` or `sizeof()` function. Pay attention to what happens to repeated values of the `$arrDistances` array.
8. Using the `foreach()` loop, output the contents of the `$arrMiles` array as a table, where the 1st column represents kilometres and the 2nd column corresponding values in miles. Format the miles to show only 3 decimal places, experiment with different type specifiers for data output. Leave space between the two columns, e.g, use the tab "`\t`" in the `printf()` function. Finally add heading to your table output.
9. Improve your code in a way that in case of repeated values in the `$arrDistances` array, the key in the `$arrMiles` array is supplemented with a letter, e.g. "A", "B", etc. Write a separate function `getKey($testKey, &$arr)`, where the parameter `$testKey` is the key to be tested and `$arr` is the array where the key is to be searched for passed by a reference. The function `getKey()` will return the key value. You can use the `array_key_exists()` function for key detection.

For PHP standard functions and their syntax, refer to http://php.net/manual/en/ and its search capabilities.

➔ Table 1 illustrates this task with sample data through various steps of this task described above.

Table 1. Example data through program run

Initial $arrDistances with random values	Sorted $arrDistances	Contents of $arrMiles	Final output	
Array	Array	Array	KM	MILES
(	(	(	12	7.456
[0] => 70	[0] => 12	[12] => 7.4564728397977	17	10.563
[1] => 20	[1] => 17	[17] => 10.563336523047	18	11.185
[2] => 21	[2] => 18	[18] => 11.184709259697	20	12.427
[3] => 66	[3] => 20	[20] => 12.427454732996	21	13.049
[4] => 18	[4] => 21	[21] => 13.048827469646	37	22.991
[5] => 12 →	[5] => 37 →	[37] => 22.990791256043 →	53	32.933
[6] => 83	[6] => 53	[53] => 32.93275504244	66	41.011
[7] => 72	[7] => 66	[66] => 41.010600618887	70	43.496
[8] => 37	[8] => 70	[70] => 43.496091565486	72	44.739
[9] => 17	[9] => 72	[72] => 44.738837038786	72	44.739
[10] => 72	[10] => 72	[72A] => 44.738837038786	83	51.574
[11] => 53	[11] => 83	[83] => 51.573937141934		
)	)	)		

Task 2 – Implement Program 2: Files, Text processing

Create a new PHP file that declares a list of vowels as follows:

```
$vowels = array("A", "E", "I", "O", "U");
```

Save the file into your lab work folder under a subfolder with a name "incl". Name the file ***phonics.php***.

Now create a subfolder "data" and a file called ***text.txt*** into it. Insert textual content into the file such that the first two lines will be in capital letters, and the rest of the document is normal text. For example:

> LOREM IPSUM
> WHAT FOLLOWS IS A TEXT OF 836 CHARACTERS.
> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin egestas tortor et nisl hendrerit efficitur. Ut ut viverra nisl. Praesent pulvinar, massa a pharetra suscipit, arcu ligula efficitur risus, lacinia rhoncus enim metus sit amet velit. Cras eu nisl ac felis vulputate accumsan in in leo. Fusce vulputate mi massa, ac viverra nisi egestas suscipit. Cras hendrerit neque turpis, nec volutpat sapien volutpat vitae. Morbi vitae ultrices est, sed placerat turpis. Etiam commodo urna fringilla, laoreet risus in, blandit ante. Quisque finibus, dui at convallis pellentesque, ex magna dapibus ligula, ut porttitor nisl tellus vitae ante. Vivamus fringilla tempus magna. Mauris ut nisl ac augue sodales mollis vitae at nisl. Ut sagittis mauris nec est porttitor, eget convallis nisl fringilla. Vestibulum molestie a odio at venenatis.

You can download this text sample from course materials, or create file similar to this on your own.

Create a new PHP file in your lab folder. Implement a program to carry out the following tasks:

1. Include the file *phonics.php* into your program file.
2. Connect (open) to the file *text.txt* (further referred as text).
3. Write a function `phonicsCount()` that counts the occurrence of all the vowels specified in the vowels list in *phonics.php*. This method returns an array with vowel-count pairs. Capital and lower-case letters are considered to be the same. The choice of function parameters for the function is up to the programmer. An example of the returned result for the sample text is given on Fig 1.
4. Write a function that returns the number of characters without whitespaces in the given text. Programmer is free to choose the name of the function and its design. For whitespace detection you can use for example `ctype_space()`.
5. Write a function that returns the number of lines in the given text. Programmer is free to choose the name of the function and its design.
6. Output the results on console.

Do note that you might need to `rewind()`/`fseek()` file pointer.

```
Array (
  [A] => 70
  [E] => 70
  [I] => 81
  [O] => 31
  [U] => 49
)
```

Figure 1. Phonics count result example for function `phonicsCount()`.