

## [IS113] Exercises - Week 3 - Basic PHP

### Objectives

- *The first set of 8 exercises are adapted from IS111 (Introduction to Programming).*
- Many students seem to require more practice to become familiar with PHP syntax.
- Revisit IS111 lab exercises - but solve it using PHP syntax.
- Simple form processing using POST and GET

### Instructions

- Questions with no asterisk mark are easy peasy.
- Questions marked with \* are slightly challenging.
- Questions marked with \*\* are challenging.
- Questions marked with \*\*\* are very challenging.

### Download

- **Resources:** Click [here](#)

**NOTE:** If you spot any mistakes/errors in the questions, please contact your instructors by email and state the issues. We will try to address it as soon as possible.

# Lab 1: Variables

Learning Outcomes:

- Solve very simple arithmetic problems by using variables referring values of different type
- Get familiar with string concatenation
- Get familiar with using print and input built-in functions

## Part A

Write a program in a file named **lab1\_1.php** that computes and prints out the area and circumference of a circle of a given radius. The radius could be set to an int or float value. You can set the value of  $\pi$  to **3.14**

**Note:** Declare  $\pi$  using a constant with name PI with all capitalized letters to indicate that the value is not to be changed in the code after the initial assignment.

Formulas:

- Area of circle =  $\pi * \text{radius}^2$
- Circumference of circle =  $\pi * 2 * \text{radius}$

Here is a sample output when **lab1\_1** is run (if radius is set to **4.0**):

```
Area of circle of radius 4 cm is 50.24 sq cm
Circumference of circle of radius 4 cm is 25.12 cm
```

## Part B

Your weight is actually the amount of gravitational attraction exerted on you by the Earth. On different planets, your weighing scales will show different figures depending on the gravitational force of that planet.

Write a program in a file named **lab1\_2.php** that defines a variable to store your weight on Earth. The program then computes and outputs your weight on Mercury, Venus, Jupiter and Saturn. Use this conversion table:

Planet	Multiply the Earth Weight by
Mercury	0.4
Venus	0.9
Jupiter	2.5

Here is a sample output of the program (if your weight on Earth is set to **60**):

```
Your weight on Earth is 60 kg
Your weight on Mercury is 24 kg
Your weight on Venus is 54 kg
Your weight on Jupiter is 150 kg
```

### **Part C**

In the code given below, complete code that circularly shifts the values of 4 variables a, b, c and d. You are NOT supposed to make use of more than one new variable to accomplish the task. For example, if the variable values (of a, b, c and d) are initially 11, 21, 31 and 41 respectively, then the final values (of a, b, c and d) are 41, 11, 21, and 31 respectively.

Note: you should not hard code the answer, which means that for any value set to variables a, b, c and d, the code written should work without any change.

```
01 <?php
02     $a = 11;
03     $b = 21;
04     $c = 31;
05     $d = 41;
06     echo "before rotation: a = $a, b = $b, c = $c, d = $d";
07
08     # write your code here
09
10
11     echo "after rotation: a = $a, b = $b, c = $c, d = $d";
12 ?>
```

Here is the output of **lab1\_4.php**:

```
before rotation: a = 11, b = 21, c = 31, d = 41
after rotation: a = 41, b = 11, c = 21, d = 31
```

### **Part D**

Write a program in a file named **lab1\_5.php** that converts a temperature reading in Fahrenheit (tempInF) to Celsius (tempInC). The formula for conversion is:

$$\text{tempInC} = (\text{tempInF} - 32) \times 5 / 9$$

- (i) Get a value from **lab1\_5.html** for temperature in Fahrenheit.
- (ii) Store the user input in tempInF, and print out the corresponding temperature in Celsius.

Here is **lab1\_5.html**:

```
<!DOCTYPE html>
<html>
  <body>
    <form action="lab1_5.php">
      Temperature in Fahrenheit:
      <input type="text" name="tempInF"/>
      <input type="submit"/>
    </form>
```

```
</body>
</html>
```

Here is a sample output when **lab1\_5** is executed (with `tempInF = 100.2`):

```
100.2 F = 37.888888888889 C
```

## **Part E**

Interest on credit card outstanding amount can be quite high. Some credit card companies compute interest on an *average daily balance*. The credit card issuer determines your average daily balance for the month by multiplying the balance you owe by the number of days you carried it, and dividing by the total number of days in the month. Here is an algorithm for computing the average daily balance and the monthly interest charge on a credit card.

- Step 1: Multiply the net balance shown on the credit card statement by the number of days in the billing cycle (i.e. number of days in the month). This is what needs to be paid.
- Step 2: Multiply the net payment received by the number of days the payment was received before the statement date. The statement date is always the last day of the billing cycle. Therefore, if payment was received on day 5 of the billing cycle, this converts to 31-5, or 26 days before the statement date. This is what has been paid.
- Step 3: Subtract the result of the calculation in step 2 from the result of the calculation in step 1.
- Step 4: Divide the result of step 3 by the number of days in the billing cycle. This value is the average daily balance.
- Step 5: Compute the interest charge for the billing period by multiplying the average daily balance by the monthly interest rate.

Here is an example. Assume the billing cycle for the month is 31 days, and the monthly interest rate is 1.33%. The credit card statement shows a previous balance of \$1,100.00. A payment of \$650.00 was made on day 19 of the billing cycle (i.e. 12 days before the statement date). The calculation of the interest charge goes like this:

- Step 1:  $\$1,100 \times 31 = \$34,100$
- Step 2:  $\$650 \times 12 = \$7,800$
- Step 3:  $\$34,100 - \$7,800 = \$26,300$
- Step 4: Average daily balance:  $\$26,300 \div 31 = \$848.39$
- Step 5: Interest charge:  $\$848.39 \times 0.0133 = \$11.28$

Write a program called **lab1\_6.php** that computes the monthly interest charged on a credit card account. Your program must get inputs from **lab1\_6.html** for the following:

- previous balance
- payment amount
- day of the billing cycle on which payment was made
- monthly interest rate

You can assume that the number of days in the billing cycle to be **31** (regardless of the month for simplicity). Choose meaningful variable names to store the values.

```
<!DOCTYPE html>
<html>
  <body>
    <form action="lab1_6.php">

      Enter previous balance:
      <input type="text" name="prev_balance"/>
      <br/>

      Enter paid amount:
      <input type="text" name="paid_amount"/>
      <br/>

      Enter day of payment made:
      <input type="text" name="day_payment_made"/>
      <br/>

      Enter interest rate:
      <input type="text" name="interest_rate"/>
      <br/>

      <input type="submit"/>
    </form>
  </body>
</html>
```

Here is a sample output for examples given earlier:

```
Previous balance is $1100
Payment of $650 was made on day 19 of the billing cycle
Interest on outstanding amount is $11.283548387097
```

## Lab 2: Conditionals and Iteration

Learning Outcomes:

- Get familiar with using if-else, if-elseif structures and boolean operators
- Get familiar with using for loop structure

### Part A

A shop pays its sales staff based on each salesperson's monthly sales. Each salesperson is paid a basic monthly salary of \$2000 plus commission based on the following table:

Monthly sales (\$)	Commission rate (%)
<10,000	5
10,000 to <15,000	10
15,000 to <18,000	15
18,000 and above	18

Complete file named **lab2\_1.php** that does the following:

1. Prompts a user to enter the monthly sales amount and submits the form's values back to itself
2. Receives the salesperson's monthly sales amount to
  - a. Calculate the salesperson's commission
  - b. Calculate the salesperson's pay (i.e. \$2000 + commission)
  - c. Display salesperson's commission rate in percentage and monthly pay.

Here is a sample output of **lab2\_1.php** (for monthly sales amount = 14450):

```
Entered monthly sales amount($):14450
Commission rate for sale of sales amount 14450 is 10%
The monthly pay for the salesperson is $3445
```

### Part B

Write a program in a file named **lab2\_2.php** that simulates a simple jackpot machine. You are given the program's partial code that generates and shows 3 random numbers (between 1 and 9, including 1 and 9) in a row.

Lines 02 to 04 calls the function in random module to generate 3 random numbers between 1 and 9 and assigns them to variables num1, num2 and num3 respectively. Line 01 enables to call the function from random module.

Complete the program to display one of the following messages: "Try again!", "2 of a kind" or "Jackpot!" depending on the 3 numbers generated.

01	<?php
02	\$num1 = rand(1,10);
03	\$num2 = rand(1,10);
04	\$num3 = rand(1,10);
05	
06	echo "***** ";
07	echo "*** \$num1 ** \$num2 ** \$num3 ** ";
08	echo "***** ";
09	
10	# write your code here
11	
12	
13	
14	?>

Here are some sample runs of **lab2\_2**: (the random numbers generated could be different)

```
*****
** 3 ** 2 ** 7 **
*****
** Try again! **
```

```
*****
** 7 ** 4 ** 7 **
*****
** 2 of a kind **
```

```
*****
** 9 ** 9 ** 9 **
*****
** Jackpot!! **
```

### **Part C**

In number theory, a perfect number is a positive integer that is equal to the sum of all its factors excluding itself. For example 6 is a perfect number because the sum of its factors i.e.  $1 + 2 + 3 = 6$

Write a program in a file named **lab2\_3.php** that takes in a positive integer, say 6, from a suitable html form. The program should then check if the number entered is a perfect number or not and print the result. You can assume that only positive integers are entered.

Some examples of perfect numbers are 6, 28, 496

Here are some sample runs of **lab2\_3**:

```
Yes, 28 is a perfect number
```

96 is not a perfect number

### **Part D**

Write a program in a file named **lab2\_4.php** that gets a binary number from a suitable HTML form (a binary number consists of 1s and 0s), and prints True or False corresponding to the digits represented by the user input, True to represent 1 and False to represent 0. You can assume that the user enters a valid binary number.

Note: You may want to use a for loop to separate the digits.

Here are some sample runs of running **lab2\_4**:

Input binary number: 1011  
True  
False  
True  
True

Input binary number: 1011011  
True  
False  
True  
True  
False  
True  
True

### **Part E**

In Singapore, personal income tax rate for residents is progressive. To make the exercise simpler, we ignore tax rebates and group all chargeable income above \$200,000 into one category with an income tax rate of 20% as shown in the table below:

Chargeable Income	Income Tax Rate (%)	Gross Tax Payable (\$)
First \$20,000	0	0
Next \$10,000	2	200
First \$30,000	-	200
Next \$10,000	3.50	350
First \$40,000	-	550
Next \$40,000	7	2,800
First \$80,000	-	3,350



Next \$40,000	11.5	4,600
First \$120,000	-	7,950
Next \$40,000	15	6,000
First \$160,000	-	13,950
Next \$40,000	18	7,200
First \$200,000	-	21,150
In excess of 200,000	20	

The actual tax rate applicable for personal income as of 2017 can be got here:

<https://www.iras.gov.sg/irashome/Individuals/Locals/Working-Out-Your-Taxes/Income-Tax-Rates/>

Write a program in a file named **lab2\_5.php** that gets input from **lab2\_5.html** for personal annual chargeable income and displays the income tax payable by the person.

Here are some sample runs of **lab2\_5**:

Input personal income: 50000  
Tax payable: \$1250

Input personal income: 80000  
Tax payable: \$3350

Input personal income: 225000.50  
Tax payable: \$26150.1

Input personal income: 300000  
Tax payable: \$41150

## **Part F**

Singapore has coins in denominations of dollar one, and cents 50, 20, 10, 5 and 1 (though 1 cent coins are not minted any more). Write a program in a file named **lab2\_6.php** that takes in a floating point value representing amount from a user (using a HTML form), for example 2.8 indicating 2 dollars and 80 cents. The program should then display the minimum number of coins required to repay the amount in coins. Assume that the user enters a value above 0 and below 10.

Here are some sample runs of **lab2\_6**:

Entered amount: 5.1  
Number of 1\$: 5  
Number of 10c: 1

Entered amount: 3.95  
Number of 1\$: 3  
Number of 50c: 1  
Number of 20c: 2  
Number of 5c: 1

### **Part G (\*)**

Write a program in a file named **lab2\_7.php** that prints ascending sequence. The program is to get inputs for starting number of ascending sequence and count of sequences (from a suitable HTML form). Assume that the user enters positive integers for all inputs.

Here are some sample runs of **lab2\_7**:

Entered count of ascending sequence:5  
Entered starting digit:2  
2 23 234 2345 23456

Entered count of ascending sequence:3  
Entered starting digit:6  
6 67 678

## Lab 3: Strings

Learning Outcomes:

- Understand how to use string methods by passing appropriate values to the methods
- Get to solve problems involving string objects

### Part A

Edit **str\_1.php** that contains a form requesting the user's name and gender so that it displays back with the surname in capital letters and the rest of the name with first letter in uppercase. **str\_1.php** should also add the prefix Mr. or Ms. to the name based on the user input for gender. Assume the following:

- The given name could be one word, or more
- Two adjacent words of the name is separated by 1 space
- If the name has 2 words, the second word is the surname
- If the name has more than 2 words, the first word is always the surname

Note: Take into consideration all possibilities of names before you start writing your code. Make use of **if-else** effectively i.e. avoid redundant use of **if-else** structure.

Hint: explore string functions `explode()`, `ucfirst()`, `strtoupper()`

Below are some sample runs:

<p>Hello Mr. TAN Wee Kiat</p> <p>Please enter your name and gender.</p> <p>Name: <input type="text" value="tan wee kiat"/></p> <p>Gender Male <input checked="" type="radio"/> Female <input type="radio"/></p> <p><input type="button" value="Submit"/></p>	<p>Please enter your name and gender.</p> <p>Name: <input type="text"/></p> <p>Gender Male <input type="radio"/> Female <input type="radio"/></p> <p><input type="button" value="Submit"/></p>
<p>Hello Ms. SUN Stefanie</p> <p>Please enter your name and gender.</p> <p>Name: <input type="text" value="Stefanie sun"/></p> <p>Gender Male <input type="radio"/> Female <input checked="" type="radio"/></p> <p><input type="button" value="Submit"/></p>	<p>Please enter your name and gender.</p> <p>Name: <input type="text"/></p> <p>Gender Male <input type="radio"/> Female <input type="radio"/></p> <p><input type="button" value="Submit"/></p>
<p>Hello Ms. Hazirah</p> <p>Please enter your name and gender.</p> <p>Name: <input type="text" value="hazirah"/></p> <p>Gender Male <input type="radio"/> Female <input checked="" type="radio"/></p> <p><input type="button" value="Submit"/></p>	<p>Please enter your name and gender.</p> <p>Name: <input type="text"/></p> <p>Gender Male <input type="radio"/> Female <input type="radio"/></p> <p><input type="button" value="Submit"/></p>

## Part B

Write a program in a file named **str\_2.php** that contains a form requesting a line of input containing an email address. **str\_2.php** then displays the *first* email address contained in that line of input.

### Assumptions:

- An email address embedded in an input line will be surrounded by an empty space at both ends.
- Each line of input will always contain at least one email address.

For example, given the input “Email: contact\_us@smu.edu.sg” as shown below, the program will extract “contact\_us@smu.edu.sg” and display it.

Below are three sample runs:

Enter a line of text with email address :

contact\_us@smu.edu.sg  
Enter a line of text with email address :

Enter a line of text with email address :

contact\_us@smu.edu.sgxyz@smu.edu.sg  
Enter a line of text with email address :

Enter a line of text with email address :

contact\_us@smu.edu.sg  
Enter a line of text with email address :

**Hint:** explore explode(), strpos()

.	.	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	
	:		c	o	n	t	a	c	t	_	u	s	@	s	m	u	.	e	d	u	.	s	g		
start_index																									
symbol_index																									
end_index																									

In order to extract the first email address in the input line, you will need to locate the boundary of the email address by obtaining the index numbers of the start\_index and the end\_index. How can you find these two index numbers? If you think carefully, you will see that start\_index marks the last space before the symbol '@' while end\_index marks the first space after the symbol '@'. Let us refer to the index of '@' as symbol\_index. Look for functions in the String module to find indices of start\_index, end\_index and symbol\_index

A special scenario is when the email address is either at the very beginning or at the very end of the input line (or both). In this case, there may not be a space before (or after) the email address. To work around this, you can first concatenate a space to both ends of the input line. Thus, if the input line is "xyz@smu.edu.sg", it becomes " xyz@smu.edu.sg ".

### Part C

**str\_3.php** contains a form that asks the user to enter a string, say text, and two characters, say start and end. Edit **str\_3.php** that searches for a substring in text such that the substring begins with the character start and ends with the character end. If there is no such substring, display the message "No such substring".

Hint: explore substr() and strpos()

Sample runs of **str\_3.php** are shown below:

Enter the string :

Enter the start char :

Enter the end char :

Substring [Fib] is found

Enter the string :

Enter the start char :

Enter the end char :

Enter the string :

Enter the start char :

Enter the end char :

Substring [bonacci s] is found

Enter the string :

Enter the start char :

Enter the end char :

Enter the string :

Enter the start char :

Enter the end char :

No such substring

Enter the string :

Enter the start char :

Enter the end char :

## Part D (\*\*)

A palindrome is a string that reads the same forward or reverse. **str\_4.php** contains a form that requests the user to enter a string. Edit **str\_4.php** so that it displays the message “The string *<input string>* is a palindrome” if the input string is a palindrome, else print the message “The string *<input string>* is not a palindrome”. **str\_4.php** should ignore digits, empty spaces or any special characters in the input string when evaluating for a palindrome.

Hint: explore preg\_replace()

Sample runs of **str\_4.php** are shown below:

<p>Enter the string : <input type="text" value="madam"/></p> <p><input type="button" value="Submit"/></p>	<p>The string madam is a palindrome</p> <p>Enter the string : <input type="text"/></p> <p><input type="button" value="Submit"/></p>
<p>Enter the string : <input type="text" value="No 'x' in Nixon"/></p> <p><input type="button" value="Submit"/></p>	<p>The string No 'x' in Nixon is a palindrome</p> <p>Enter the string : <input type="text"/></p> <p><input type="button" value="Submit"/></p>
<p>Enter the string : <input type="text" value="bad, chocolate dab"/></p> <p><input type="button" value="Submit"/></p>	<p>The string bad, chocolate dab is not a palindrome</p> <p>Enter the string : <input type="text"/></p> <p><input type="button" value="Submit"/></p>

## Part E

**str\_5.php** contains a form that asks the user to enter two strings. Let us call the first string str1 and the second string str2. Edit **str\_5.php** such that it displays "Bingo!" if every character in str1 also appears in str2. Otherwise, it displays "Nope :( "

Hint: explore strlen(), strpos()

Some sample runs of **str\_5.php** are shown below:

<p>Enter characters: <input type="text" value="daily"/></p> <p>In string? : <input type="text" value="Holiday"/></p> <p><input type="button" value="Submit"/></p>	<p>Bingo!</p> <p>Enter characters: <input type="text"/></p> <p>In string? : <input type="text"/></p> <p><input type="button" value="Submit"/></p>
<p>Enter characters: <input type="text" value="lily"/></p> <p>In string? : <input type="text" value="Holiday"/></p> <p><input type="button" value="Submit"/></p>	<p>Bingo!</p> <p>Enter characters: <input type="text"/></p> <p>In string? : <input type="text"/></p> <p><input type="button" value="Submit"/></p>
<p>Enter characters: <input type="text" value="pokemon"/></p> <p>In string? : <input type="text" value="pogd"/></p> <p><input type="button" value="Submit"/></p>	<p>Nope :(</p> <p>Enter characters: <input type="text"/></p> <p>In string? : <input type="text"/></p> <p><input type="button" value="Submit"/></p>

## Lab 4: Functions

Learning Outcomes:

- Understand how to write your own functions and invoke them
- Know the difference between invoking a built-in function and function of an imported module

### Part A

**fun\_1.php** contains a form that accepts two integers, say  $n$  and  $f$ . Edit **fun\_1.php** such that it displays whether  $f$  is a factor of  $n$  or not. You are required to implement a function called `is_factor` in **fun\_1.php** that takes in 2 parameters  $n$  and  $f$  and returns True if  $f$  is a factor of  $n$  and False otherwise.

Assume that the user provides only integer values.

Here are the sample runs:

Enter an integer n:   
Enter another integer f:

3 is not a factor of 10

Enter an integer n:   
Enter another integer f:

Enter an integer n:   
Enter another integer f:

2 is a factor of 10

Enter an integer n:   
Enter another integer f:

### Part B

**fun\_2.php** contains a form that accepts two integers, say  $m$  and  $n$ . Edit **fun\_2.php** such that it displays the sum of the powers from  $m^0$  to  $m^n$  (i.e,  $m^0 + m^1 + m^2 + \dots + m^n$ ). In **fun\_2.php**, you are required to define a function named `sum_of_powers` that takes in 2 parameters  $m$  and  $n$ . and computes the sum of the powers from  $m^0$  to  $m^n$  and returns the sum.

Hint: explore `pow()`

Assume that the user provides only integer values.

Here is a sample run:

Enter an integer m:   
Enter another integer n:

Sum of all powers is 31

Enter an integer m:   
Enter another integer n:

### Part C

**fun\_3.php** contains a form that accepts two integers, say  $min$  and  $max$ . Edit **fun\_3.php** such that it displays all the perfect squares between  $min$  and  $max$ . For example, given  $min=10$  and  $max=110$ , it displays the perfect squares – 16 25 36 49 64 81 100.

Assume that the user provides only integer values.

Hint: explore `sqrt()`, `floor()`, `ceil()`, `pow()`

Here is a sample run:

16 25 36 49 64 81 100

Enter an integer min:   
Enter another integer max:

Enter an integer min:   
Enter another integer max:

### Part D (\*)

A sandwich typically consists of cheese, meat or/and vegetables placed in between two slices of bread. Consider a string “remember”, the sub-string “memb” is sandwiched between “re” and its mirror “er” on the other side. In **fun\_4.php**, write a function named `get_sandwich` that takes in a string and returns the sandwiched string, if any, or None.

Hint: explore `substr()`

Here are some sample outputs when **fun\_4.php** is run:

Sandwich substring for remember: memb

Enter a sandwich string:

Enter a sandwich string:

Sandwich substring for foolproof: lpr

Enter a sandwich string:

Enter a sandwich string:

Sandwich substring for oops: None

Enter a sandwich string:

Enter a sandwich string:

### Part E (\*\*\*)

You may have written a program earlier using Caesar Cipher encoding, in which each letter is replaced by the letter that is some fixed positions away from the original letter.

Let us use another simple form of encoding wherein each letter is replaced by its hexadecimal form (<https://www.wikihow.com/Understand-Hexadecimal>). Each of the a-z alphabets, digits 0-9 and special characters ( like , or . or space or ! ) are converted to a hex form using UTF-8 encoding as seen in ASCII table. There are many Text to Hex converters online. Example: <https://www.online-toolz.com/tools/text-hex-converter.php>. You can test your encoding program using the online converter.

In **fun\_5.php** define two functions called `encode` and `decode`. The function `encode` takes in a string and returns the encoded string in hexadecimal format. The function `decode`, on the other hand takes in the encoded string in hexadecimal format and returns the original string.

You can assume that the string passed into the function `encode` contains only valid characters from a-z or A-Z and punctuation symbols. You can also assume that a valid encoded string is passed into the function `decode`, i.e. every 2 characters represent the hexadecimal form corresponding to the original character.

Here are sample outputs when **fun\_5** is run:



Encoded string for Hello: SGVsbG8=

Enter a string:

Enter a string:

Decoded string for SGVsbG8=: Hello

Enter a string:

Enter a string:

## Lab 5: Indexed Arrays

Learning Outcomes:

- Get familiar with using indexed array
- Get to know to modify arrays, add, remove elements from arrays
- Get to solve problems using arrays

### Part A

Examine **arr\_1.php** given to you. Complete the function `get_numbers` to return a new array that contains only integers between `min` and `max` parameters, inclusively. Your code should not modify the original array.

Here is a sample output when **arr\_1.php** is run (with the given test code):

```
Original array: [4, 10, 12, 28, 24, 18, 5, 20, 15, 21, 30, 22, 21, 14, 17, 28, 26, 24, 6, 8, 15]
After function is called [10, 12, 18, 20, 15, 14, 17, 15]
```

### Part B (\*)

**arr\_2.php** contains a form that gets 10 integer inputs from the user. It should then display the minimum, maximum and median of all numbers entered.

Note: The median is the middle of list of numbers. For example, median of numbers 12, 4, 5 is 5. In case of odd amount of numbers, the median is the exact middle number of numbers when arranged sorted. In case of even amount of numbers, we would get a pair of middle numbers. The median is half way between them. As an example, median of numbers 6, 12, 4, 10 is 8 ( $(6 + 10) / 2$ ) because when placed in order the middle numbers would be 6 and 10.

Hint: explore `sort()`

Here is a sample run:

Enter 10 integers (separate each integer with a space):	<input type="text" value="-9 4 -11 19 5 17 4 2 8 1"/>	Sorted numbers: -11 -9 1 2 4 4 5 8 17 19
<input type="button" value="Compute Stat"/>		Minimum of numbers: -11
		Maximum of numbers: 19
		Median of numbers: 4
	Enter 10 integers (separate each integer with a space):	<input type="text" value="6 4 1 1"/>
	<input type="button" value="Compute Stat"/>	

### Part C

In **arr\_3.php** write a function called `count_numbers` that takes in a multidimensional array, say `$numbers`, containing numbers and arrays that contain numbers. It returns the count of numbers in the array `$numbers`. Note that your function has to cater to the possibility of `$numbers` having an array of numbers. You can assume that it is at the most a 2-dimensional array.

Hint: explore `is_array()`

Here are some examples:

```
$numbers = [4,6,[1,2],10,[-1,-3]];
Count of numbers:7

$numbers = [4, 6, [1,2,3,4], 10, [-1,-3], [5,7,1,2]];
Count of numbers:13
```

## Lab 6: Loop structure (any loop structure is allowed)

Learning Outcomes:

- Get familiar with using loop structure
- Get to solve problems using functions, conditional structure, for, foreach or while loop structure

### **Part A (\*\*)**

Counting in binary is similar to counting in any other number system. Beginning with a single digit, counting proceeds through each symbol, in increasing order. Decimal (or base-10) counting uses the symbols **0** through **9**, while binary only uses the symbols **0** and **1**. Read more about how a decimal number can be converted to its binary equivalent:

- <http://www.is.wayne.edu/olmt/binary/page3.htm>
- <http://www.wikihow.com/Convert-from-Decimal-to-Binary>

Let us see the method to convert decimal number 32(base 10) to its binary equivalent.

32 divided by 2 gives 16 and remainder **0**  
16 divided by 2 gives 8 and remainder **0**  
8 divided by 2 gives 4 and remainder **0**  
4 divided by 2 gives 2 and remainder **0**  
2 divided by 2 gives 1 and remainder **0**  
1 divided by 2 gives 0 and remainder **1**

Do you notice that you are dividing the given decimal number by 2 and subsequently in every iteration, the quotient is divided by 2. The binary number is the sequence of remainders in reverse, from the bottom remainder to the top remainder.

Write a program in a file named **lab6\_6.php** that converts a positive integer number (received from a HTML form) to its binary equivalent following the procedure explained above. You can check if your conversion is correct using the built-in function.

Try the built-in function `decbin` in PHP that converts decimal number to its binary equivalent.

Example #1 `decbin()` example

```
<?php
echo decbin(12) . "\n";
echo decbin(26);
?>
```

The above example will output:

```
1100
11010
```

From: <http://php.net/manual/en/function.decbin.php>

Sample run of the program is shown below:

Input decimal number is 32  
Binary equivalent of 32 is 100000

### **Part B (\*\*\*)**

Define a function called **merge()** that merges two associative arrays containing key-value pairs representing people. Each key-value pair in the association array represents one person's name and age and the list is sorted in increasing order of people's age.

The function merges the two lists into a single list in which the people are still ordered by their age.

Note: You are NOT allowed to use any sort function.

For example, if

```
$list1 = ["John"=>12, "Kate"=>15, "Henry"=>35], and  
$list2 = ["Mike"=>18, "Scott"=>20, "Joseph"=>48, "Larry"=>54]
```

When invoked like this:

```
var_dump(merge($list1, $list2))
```

The output would be:

```
array (size=7)  
'John' => int 12  
'Kate' => int 15  
'Mike' => int 18  
'Scott' => int 20  
'Henry' => int 35  
'Joseph' => int 48  
'Larry' => int 54
```

**Note:** You are allowed to make your own assumptions

## Lab 8: Dictionaries (Associative Arrays)

### Part A (\*)

Complete the function `compute_bill` in the file that takes in a dictionary of items purchased by the customer, and price of items and returns the amount to be paid.

```
01 <?php
02 $price_info = [
03     'pencil'=> 0.80,
04     'pen' => 1.20,
05     'eraser' => 0.50
06 ];
07
08 #complete this function
09 function compute_bill($cart, $pricing){
10
11 }
12
13 $jane_items = ['pen'=>10, 'eraser'=>2];
14 $eric_items = ['pencil'=>12, 'eraser'=>5, 'pen'=>2];
15
16 echo "Jane's bill amount $" . compute_bill($jane_items, $price_info);
17 echo "Eric's bill amount $" . compute_bill($eric_items, $price_info);
18 ?>
19
20
21
22
```

Here is the expected output (with the given test data):

```
Jane's bill amount $ 13
Eric's bill amount $ 14.5
```

### Part B (\*\*)

4. [\*\*] In a file named **lab8\_4.php**, copy the code shown below. Write a function called `reverse_dict` in the file that takes in a dictionary and reverses the dictionary. That is, you are supposed to return a new dictionary that creates keys out of values and values out of keys. You can assume that the input to the functions consists of dictionary whose values are indexed arrays (aka. lists).

```

01 <?php
02 #write the function
03
04 $dict1 = reverse_dict(["a"=>[1,2,3], "b"=>[1,2], "c"=>[3,4], "d"=>[5,6]]);
05 $student_subjects = reverse_dict(["Jane"=>["Economics","Physics","Chemistry"],
06     "Mark"=>["Literature","Chemistry","Biology"],
07     "Sarah"=>["Literature","Physics","Chemistry"]]);
08
09 var_dump($dict1);
10 var_dump($student_subjects);
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27 ?>
28

```

Here is the output when **lab8\_4** is run (with the given two test data):

C:\wamp64\www\is111\lab8\ex4.php:5:

```

array (size=6)
  1 =>
    array (size=2)
      0 => string 'a' (length=1)
      1 => string 'b' (length=1)
  2 =>
    array (size=2)
      0 => string 'a' (length=1)
      1 => string 'b' (length=1)
  3 =>
    array (size=2)
      0 => string 'a' (length=1)
      1 => string 'c' (length=1)
  4 =>
    array (size=1)
      0 => string 'c' (length=1)

```

5 =>

**array** (size=1)

0 => string **'d'** (length=1)

6 =>

**array** (size=1)

0 => string **'d'** (length=1)

C:\wamp64\www\is111\lab8\ex4.php:6:

**array** (size=5)

'Economics' =>

**array** (size=1)

0 => string **'Jane'** (length=4)

'Physics' =>

**array** (size=2)

0 => string **'Jane'** (length=4)

1 => string **'Sarah'** (length=5)

'Chemistry' =>

**array** (size=3)

0 => string **'Jane'** (length=4)

1 => string **'Mark'** (length=4)

2 => string **'Sarah'** (length=5)

'Literature' =>

**array** (size=2)

0 => string **'Mark'** (length=4)

1 => string **'Sarah'** (length=5)

'Biology' =>

**array** (size=1)

0 => string **'Mark'** (length=4)

## Question 9 - PHP

Learning Outcomes:

- Get familiar with PHP associative array
- Practice displaying content using HTML table

Go to **php** directory.

Complete the following **Parts A, B** and **C** inside **practice.php** file.

**NOTE:** Please do not hard-code the names and the grades. You must retrieve it from the associative array.

**HINT:** You must use a loop in PHP.

### Part A

Given an associative array as shown below:

```
$student_grades = [  
    'Kee Hock' => ['A+', 'A', 'B+', 'A-'],  
    'Debbie' => ['A', 'B+', 'A-', 'A'],  
    'Patrick' => ['B', 'C', 'F', 'B-']  
];
```

Complete **Part A** in the **practice.php** file so that it displays the following HTML table.

Name	Grades
Kee Hock	A+
Kee Hock	A
Kee Hock	B+
Kee Hock	A-
Debbie	A
Debbie	B+
Debbie	A-
Debbie	A
Patrick	B
Patrick	C
Patrick	F
Patrick	B-



### Part B (\*)

Given the same associative array, complete **Part B** in the **practice.php** file so that the following HTML table is displayed. Please take note of the following:

- If the grade is **A+** or **A**, font color should be **blue** and font size should be **6**.
- If the grade is **A-**, font color should be **green** and font size should be **5**.
- Otherwise, for all other grades, font color should be **red** and font size should be **4**.

Name	Grades
Kee Hock	A+
Kee Hock	A
Kee Hock	B+
Kee Hock	A-
Debbie	A
Debbie	B+
Debbie	A-
Debbie	A
Patrick	B
Patrick	C
Patrick	F
Patrick	B-

### Part C (\*\*)

Given the same associative array, complete **Part C** in the **practice.php** file so that it displays the following HTML table.

Name	Grades
Kee Hock	A+
	A
	B+
	A-
Debbie	A
	B+
	A-
	A
Patrick	B
	C
	F
	B-

## Question 10 - Vegetables

Learning Outcomes:

- Practice displaying drop-down menu using HTML
- Get familiar with form submission via POST and GET

Go to **veg** directory. It contains the following:

Item	Description
<b>index.php</b>	1. An associative array <b>\$vegTypeArr</b> whose key is vegetable name and value is the type of vegetable, and 2. An incomplete form that will submit to <b>process.php</b> .
<b>process.php</b>	An associative array <b>\$vegQtyArr</b> whose key is vegetable name & value is the quantity available.
Folder <b>img/</b>	Contains JPEG images of leafy vegetables: <b>cabbage.jpg</b> , <b>lettuce.jpg</b> and <b>spinach.jpg</b> .

### Part A (\*)

1. Edit **index.php** to add a dropdown list of **leafy** vegetables
  - a. Get the vegetable names from the array **\$vegTypeArr**.
  - b. Do NOT hardcode the vegetable names.
2. If done correctly, the form should look like this:

Leafy Vegetable 

spinach ▾  
cabbage  
lettuce  
spinach

### Part B (\*)

1. Edit **process.php** to
  - a. Variable **\$veg** is the selected vegetable (submitted form value).
  - b. Obtain the quantity from **\$vegQtyArr**
  - c. Display the image of the vegetable as many times as the quantity.
2. For example, if the selected vegetable is 'spinach', there are 2 spinaches, thus, the page should look like this.

**spinach**



## Question 11 - Password Validation

Write the relevant codes to perform password validation with the following password rules:

- \$password and \$confirmPassword must be the same
- must not start and end with the same character (e.g. "abcdefga")
- must not contain the same character more than 3 times (e.g. "abcdefgggg ")

You should display the relevant error message(s) if the password violates the rules or display a message "The password is valid." if it meets all the rules.

```
<!DOCTYPE html>
<html>
<body>
<?php
    $password = "abcdefgh";
    $confirmPassword = "abcdefgh";

    //write your codes here, change the value of the $password and $confirmPassword fields
    to test for the rules accordingly.

?>
</body>
</html>
```

## Question 12 - HTML Form

### Part A

Write the relevant codes to generate HTML elements dynamically based on the following logic. The \$numUsers field will determine the number of users that we would need to capture in the form. For each user, the following HTML controls should be generated.

E.g. if the value of \$numUsers is 2, the output should be:

#### User 1

Name :

Password :

#### User 2

Name :

Password :

```
<!DOCTYPE html>
<html>
<body>
<form>
<?php
    $numUsers = 2;

    //write your codes here, change the value of the $numUsers to test

?>
</form>
</body>
</html>
```

### **Part B (\*)**

Add in the relevant attributes for the password HTML elements such that the user can only key in a maximum of 12 characters.

For every alternate user (e.g. 2nd, 4th, 6th .. user), add another field to capture the age

#### **User 1**

Name :

Password :

#### **User 2**

Name :

Password :

Age :

#### **User 3**

Name :

Password :

#### **User 4**

Name :

Password :

Age :

## Question 13 - Sales Table

### Part A

The following table represents the quarterly sales amount for a company over a 5 years period. Represent this as a two-dimensional array. Find the **average** sales amount of each year.

	Q1	Q2	Q3	Q4
Year 1	1000	750	500	900
Year 2	200	600	700	800
Year 3	300	650	780	890
Year 4	600	700	800	900
Year 5	670	550	500	700

The output of the resultant php page should be as follows:

Average Sales in Year 1: 787.5

Average Sales in Year 2: 575

Average Sales in Year 3: 655

Average Sales in Year 4: 750

```
<!DOCTYPE html>
<html>
<body>
<?php
    $sales_arr = ...; // complete me and the rest of the code!
?>
</body>
</html>
```

### Part B (\*\*)

Compute medians, modes, standard deviations, and variances in addition to averages. Display the metrics as an unordered list.

## Question 14 - Courses

Represent the table below as an associative array of indexed array (aka. a dictionary of lists):

Course1	Bob, Ann, Anthony, Andrew
Course2	Bob, John, Cain
Course3	Jane, Jill, Cain
Course4	Bob, Jane

Then, compute the total number of courses each person registers. The output should be:

Bob registers for 3 courses  
Ann registers for 1 courses  
Anthony registers for 1 courses  
Andrew registers for 1 courses  
John registers for 1 courses  
Cain registers for 2 courses  
Jane registers for 2 courses  
Jill registers for 1 courses

```
<!DOCTYPE html>
<html>
<body>
<?php
    $course_registration = ...; // complete me and the rest of the code!
?>
</body>
</html>
```

## Question 15 - Taxis

Represent the table below as an indexed array of associative arrays (aka. a list of dictionaries):

0	Taxi1=50, Taxi2=25, Taxi3=27
1	Taxi1=40, Taxi2=34, Taxi3=17
2	Taxi1=60, Taxi2=25
3	Taxi1=10, Taxi3=47

Then, compute compute the total number of trips each taxi makes. The output should be:

Taxi1 made 160 trips  
Taxi2 made 84 trips  
Taxi3 made 91 trips

```
<!DOCTYPE html>
<html>
<body>
<?php
    $taxis = ...; // complete me and the rest of the code!
?>
</body>
</html>
```



## Question 16 - Sudoku

### Part A

The following table represents a partial sudoku puzzle. Represent this as a two-dimensional array. Check that the 3 x 3 box contains the digits 1 through 9 by completing the function **isValid**. Additionally, complete function **printBox** to get a nice visual output of a sudoku box.

1	4	5
6	3	9
8	2	7

The output of the resultant php page should be as follows:

The following sudoku box

1	4	5
6	3	9
8	2	7

is **valid**

A partially completed php page is shown below:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    table,th,td{
      border: 1px solid black;
      border-collapse: collapse;
    }
  </style>
</head>
<body>
<?php
  $sudoku_box = ...; // complete me and the rest of the code!

  echo "The following sudoku box <br/>";
  printBox($sudoku_box);

  if (isValid($sudoku_box)){
    echo "is <strong>valid</strong>";
  }
  else {
```

```
    echo "is <strong>Invalid</strong>";
}

function printBox($sudoku_box){
    // oh no i'm empty, complete me!
    // but you may declare other functions as you wish - don't have to use me!
}

function isValid($sudoku_box){
    // oh no i'm empty, complete me!
    // but you may declare other functions as you wish - don't have to use me!
}
?>
</body>
</html>
```

### **Part B (\*\*\*)**

Extend the code so that an explanation is given when a sudoku box is invalid (e.g., "There is less than 3 rows", "Number 9 is missing from the box", etc.). Extend the code so that isValid accepts a full sudoku and not just one of its nine sub boxes (<https://en.wikipedia.org/wiki/Sudoku>).