Lab 1 - Functions

Warm up mini-lab

Type in the following function definition and calls to the function. Test it and understand it.

```php
function myprint($str="No String Supplied") { 
    echo "<br><h1>".$str1."</h1>"; // What will be printed
    $len=strlen($str); // The strlen() function returns the length of the argument
    return $len;
} // end of function definition

// the following lines test the function

$length=myprint(); $defaultstr ="No String Supplied"; $stringarg="The is a new string argument";
length=myprint($stringarg); //test with a non-empty string

<$br>The length of $defaultstr is $length characters
The length of $stringarg is $length characters;
```

Lab Exercise 1

Create a function called 'arith' which takes three arguments: a string and two numbers.

The string argument is expected to be one of the following strings specifying arithmetic operations:

"add", "sub", "mult", "div".

The function returns the result of performing the given operation on the two number arguments.

E.g. $result=arith("sub", 8, 10); will set $result to be equal to -2.

The numeric arguments are optional; If either numeric argument is missing, 0 (zero) will be assumed.
The function should use a switch statement to test the string argument,

Your script must test for division by zero at the appropriate point. If this condition is detected the script must return a value of 0 and terminate with an error message.

The Default case must be used in the case where the user types an illegal string;
You must provide descriptive comments for each major block of code in your script.

Use the built-in function strtolower() to allow users to type in the operator arguments in any case (upper, lower, or mixed).

Following your function definition, provide several call to the function. Try to test all the unusual conditions that a user of your function might inadvertently (or deliberately) try.
Lab 2 - Arrays

1. Using nested **for loops**, initialize a **two-dimensional array** with the 12 x 12 multiplication table.

   Use an **indexed array** (not associative) and the **array operator**, [], rather than the array function to accomplish this task.

   Again using nested for loops, **display the array contents**. You do not have to use a <table>; the format in the following output fragment is acceptable

   
   |   |   |   |   |   |   
   |---|---|---|---|---|---|
   | 1 | 2 | 3 | 4 | ... | 12 |
   | 2 | 4 | 6 | 8 | 24 |   |
   |   | 12| 24| 36| 48 | ... |

2. Using the appropriate array function, display the **size** of the array.

3. Create a two-dimensional associative array, which describes part of your music collection. The outer array has as many inner arrays as you want (at least 2); the inner arrays each have three keys: $title, $artist, $media. (Values for media would be “CD”, “LP”, “Cassette”; $artist and $title are the performer and the name of the album, respectively. Use nested foreach() loops to print out all keys and values for albums.
1. a. Set 3 variables exactly as follows (you may use your own variable names.):
   
   $\text{var1} = 27; \quad \text{var2} = 15.50; \quad \text{var3} = \text{"Britney Spears"}$

   b. Write printf() statement that prints the following line, exactly as shown.:
      You must use only variables in your code, no values.
      
      Our group paid $15.50 each for 27 tickets to see B. Spears.

2. Write an echo statement that prints the following line exactly as shown:
   
   Today is the 27th day of September, in the year 2007.
Laboratory Exercise #4 - Forms

1. Write an html page containing a form which will provide the arguments for the function written in Laboratory Exercise 1. The function is executed and the results of the operation are displayed, with some descriptive text.

Recall that this exercise involved writing a function to take two numerical arguments and one string argument (an arithmetic operation) and returned the result of performing the arithmetic operation on the two numbers.

2. Use text boxes for the numerical input.

3. Use a `<select>` html element for the arithmetic operator input.

4. The function definition (from Lab 1) must be on the same page as the form.

This is so the user can continually enter new numbers and operators and obtain new answers. Note that no modification of the function itself is required, provided you choose the name attributes in your form appropriately.

Lab 1:
Create a function called 'arith' which takes three arguments: a string and two numbers. The string argument is expected to be one of the following strings specifying arithmetic operations: "add", "sub", "mult", "div". The function returns the result of performing the given operation on the two number arguments.
E.g. $result=arith("sub", 8, 10); will set $result to be equal to -2.
The numeric arguments are optional;
If either numeric argument is missing, 0 (zero) will be assumed.
The function should use a switch statement to test the string argument,
Your script must test for division by zero at the appropriate point. If this condition is detected the script must return a value of 0 and terminate with an error message.
The Default case must be used in the case where the user types an illegal string;
You must provide descriptive comments for each major block of code in your script.
Following your function definition, provide a call to the function. Try to test all the unusual conditions that a user of your function might inadvertently (or deliberately) try.
Laboratory Exercise #5 - Cookies

Create three .php pages, one to set a cookie, one to read the cookie, and one to destroy the cookie.

1. **On the page which sets the cookie:**

   Use the `setcookie()` function with the following parameters:

   - Cookie name of your choice
   - Cookie value of your choice
   - Expiration date equal time()+3600
   - The path and domain may be left blank

   **Put a link** to the page which **reads** the cookie.

2. **On the page which reads the cookie**

   Use an **if statement** to test whether the cookie has been set.
   
   If it has, **display all the available parameters** along with descriptive text.
   
   If no cookie is set, **print an appropriate message**.

   **Provide a link** to the page which **destroys** the cookie.

2. **On the page which destroys the cookie**

   **Destroy** the cookie.

   **Provide a link** to the page which **sets** the cookie.
Files warm-up Lab 1

The following script displays the contents of 'labfile.txt' on this web page using file().</h3>

```php
<?php
if (file_exists("labfile.txt"))
{
    $filearray=file("labfile.txt");
    foreach ($filearray as $line)
    {
        echo $line."<br>";
    }
    echo "<br>The size of labfile.txt is ".filesize("labfile.txt");
} else { echo "<br>No such file."; }
?>
```

Files warm-up Lab 2

The following script displays the contents of 'labfile2.txt' on this web page using fgets().</h3>

```php
<?php
if (file_exists("labfile2.txt"))
{
    $file=fopen("labfile2.txt", "r");
    While (!feof($file))
    {
        echo fgets($file, 1024)."<br>";
    }
    rewind($file);
    fclose($file);
    echo "<br>The size of labfile2.txt is ".filesize("labfile2.txt");
} else {echo "<br>No such file."; }
?>
```
Laboratory Exercise #6 - Files

1. Write a form which has two text input boxes and one password box. The two text boxes are for the first and last names of the user, the password box is for their password.

2. Write a php script which
   a. Checks for the existence of the password file
   b. If it does not exist, create it and write the three form variables to the file. Display a message indicating that no password exists.
   c. If the file does exist, read the data and compare it with the data sent from the form. Display a message stating whether or not all three pieces of data matched the contents of the file.
   d. If the file does exist, and neither the first nor last name match the contents of the file, append the new data to the file

3. Close the file.

4. Display the contents of the file. (not a good idea in real life. Also, in real life we would encrypt the password.)

HINT: Use strtolower() to store the data and to match the form data to the stored data.
Laboratory Exercise #7 - MySQL

1. **Write a script** that displays the results of a SELECT query.

   Note that the functions used in the following script differ somewhat from what is in the book.

   ```php
   $conn=mysql_connect (“localhost” “dbo142755164”, “FrnK3Exe”,);  // connect to mysql
   mysql_select_db(“db142755164”, $conn) or die(mysql_error()); //select db
   $querystr=“SELECT * FROM Users”; //create query
   $resultptr=mysql_query($querystr, $conn) or die(mysql_error()); //execute query
   $number_of_records=mysql_num_rows($resultptr);
   for ($record=0; $record<=$number_of_records-1; $record++) //display
   {
     $field_array=mysql_fetch_array($resultptr); //returns record as an array of fields
     echo $field_array[‘username’]. “ ”;
     echo $field_array[‘password’]. “ ”;
     echo $field_array[‘nameFirst’]. “ ”;
     echo $field_array[‘nameLast’].”<br>“;
   }
   mysql_close($resultptr);
   ```

2. Write queries to do the following

   a. **Display all the records** in a table
   b. **Display a subset (using the WHERE clause) of the records** in the table.
   c. **Add (INSERT) a record** into the table and display the result
   d. **Delete a record** from the table and display the result.

For this exercise you will use an existing table in an existing data base.

When you connect to the database use ‘localhost’ as the host name and
**dbo142755164** as the userid; the password is. **FrnK3Exe**.

The name of the table is **Users** (case-sensitive)

The table has four fields called: **username, password, nameFirst, nameLast**, all of which are of type **varchar(25)**
DO NOT INCLUDE FOLLOWING WHEN PUBLISHING THE LAB

Script modified for part b above

```php
$conn=mysql_connect("localhost" "dbo142755164", "FrnK3Exe");  // connect to mysql
mysql_select_db("db142755164", $conn) or die(mysql_error()); //select db
$querystr="SELECT nameFirst, nameLast, FROM Users WHERE nameFirst = 'Alice'";  //create query
$resultptr=mysql_query($querystr, $conn) or die(mysql_error()); //execute query
�数量_of_records=mysql_num_rows($resultptr);
for ($record=0; $record<=$数量_of_records-1; $record++) //display results
{
    $field_array=mysql_fetch_array($resultptr); //returns record as an array of fields
    // echo $field_array['username']." ";
    // echo $field_array['password']." ";
    echo $field_array['nameFirst']." ";
    echo $field_array['nameLast']."<br>";
}
mysql_close($resultptr);
```
Lab #8 - MySQL Table creation

This exercise gives you experience in creating tables for a MySQL data base. **You will use this table in Homework #6.** The table design is as follows

Each record in the table will have four fields:

- An auto-incrementing index field called `userid`, of type INT
- A field for the user’s last name called `lname` of type varchar(30)
- A field for the user’s first name called `fname` of type varchar(15)
- A field for the user’s password called `passwd` of type varchar(40)

Connect to the MySQL server and select the data base as in lab#6. The user id which you will use to connect to the mysql server is `dbo181508648`, the host name is `localhost`. The password =G6.YB8P4

Send the query to the MySQL server using

```
mysql_query($createtable, $conn) or die(mysql_error());
```

In this case the mysql_error() function is very useful since you have no other way of knowing whether the table was successfully created.

Insert at least **two different records** into the data base.
Lab #9 - MySQL Table Initialization

This lab initializes the table you will be using in HW #7. You will also find it useful for your project.

Write a form that

a. asks the user for a **password, last name, and first name** (use three text input boxes). No input is required for the **userid** since that is an auto-incrementing integer, a primary key.
   
   b. Includes three radio buttons so that the user can choose to **add** a record, **delete** a record, or **view** the contents of the entire table

Write a script, on the same page, that does the following:

1. Checks that the form has been submitted
2. Checks that all three boxes have had something typed into them (validation).
   
   a. If not, print an error message and return to the form
   
   b. If so, assign all of the $_POST variables to local variables
3. Connect to the MySQL server, using **userid dbo181508648** and the password given in class, and select **database db181508648. pw = G6.YB8P4**
4. Form the SQL query string $querytstr, which will contain either a SELECT query; an INSERT query, or a DELETE query, as determined by which radio button was selected. (Use the code from Lab #5 to display the records when the **view** button is pressed.)

   Remember that the **table** you are using is named **xypasswords**, where x and y are your **initials** as defined in HW#6

   Note: This will be a little tricky since the **INSERT sql syntax requires that the data be in single quotes**, but you will be using the string variables derived from the form. We will cover this in class.

   Also note that the columns to have values must be explicitly specified since not all columns are being set by form data; specifically, the userid is being set automatically by auto-increment.

5. Execute the query and return to the form so that the user may add another entry.
Lab #10 - Validation and Redirection

This lab gives you experience in validating form input and shows you how to redirect the user to another page (note that this is required for the project, as well!). You will create two pages.

First Page (lab6.php)

1. Write a form that contains a password box and a text box for input. Tell the user that the password box must contain a string of at least 8 characters in length, and the other box must contain a number between 20 and 90, inclusive.

2. Write a script, on the same page, which performs the following tasks:
   a. **Test whether the submit button** has been pressed. If not, do nothing.
   b. If the submit button has been pressed, write code which **tests that**
      i. neither box has been left empty
      ii. The password box has a string length of 8 characters
      iii. The numeric value is in the required range.
   c. If all of the above conditions have been satisfied, **redirect** the user to the other page: lab6.htm. Otherwise, output an appropriate error message and **return the user to the form**. Remember that the header() function must appear before any html, if any exists on the page.

Second Page (lab6.htm)

This page simply has **a message of welcome and congratulations**. No scripting is required.