	UNIVERSIDAD DON BOSCO **FACULTAD DE INGENIERÍA** **ESCUELA DE COMPUTACIÓN**
CICLO I-2021	**Lenguajes Interpretados en el Servidor** **Guía de práctica No. 6** **Funciones con PHP**

I. OBJETIVOS

- Adquirir el dominio de la sintaxis y el uso de funciones con el lenguaje PHP.
- Lograr la habilidad necesaria para la declaración de funciones con o sin argumentos y con devolución o no de valores.
- Hacer uso de características avanzadas de las funciones como la lista variable de argumentos, las funciones variables y recursividad.
- Hacer uso de la modularidad para crear scripts que serán incluidos dentro de un script principal.

II. INTRODUCCIÓN TEÓRICA

Funciones

Una función es un bloque de código independiente y autónomo que contiene un grupo de instrucciones que se identifican con un nombre. Las funciones pueden invocarse todas las veces que se requiera desde cualquier punto de un script. El código de las funciones puede aparecer dentro del script que se esté realizando o puede ser parte de un script independiente que será llamado como archivo de inclusión (otro script PHP invocado).

Generalmente, cuando se realizan scripts con PHP las instrucciones se ejecutan conforme van siendo procesadas por el intérprete del lenguaje. Sin embargo, a la hora de realizar scripts más complejos, hay ocasiones en las que el mismo código ha de ser ejecutado más de una vez. Para estos casos sería útil que el lenguaje de programación permitiera dividir el código en segmentos más pequeños, de forma que cada bloque de código PHP pudiera ser agrupado bajo un identificador para que pudiera ser accedido de forma independiente.

PHP permite la creación de funciones que permiten recoger todos los aspectos planteados anteriormente. Una función permite desarrollar una tarea concreta y bien definida. Se encuentra separada del resto de instrucciones del programa y se le asigna un nombre para que posteriormente pueda ser llamada desde otro punto del script, o incluso, desde otro script. Es a través del nombre de la función que se pueden ejecutar las instrucciones contenidas en la función tantas veces como sea necesario. La utilización de funciones permite que un script aparezca escrito como una lista de referencias a las tareas que se deben hacer para crear una página de respuesta.

Las funciones también pueden o no, aceptar parámetros o argumentos externos de los que dependa el proceso que deba realizarse para posteriormente devolver un valor.

Sintaxis de las funciones en PHP

En su forma más compleja una función se declara con la palabra reservada function. A continuación, se debe declarar el nombre de la función y posteriormente, y entre paréntesis, cada uno de los argumentos que recibirá separados por coma. En el caso más simple, sólo se colocan los paréntesis vacíos inmediatamente después del nombre de la función. Después de cerrar los paréntesis debe abrir llaves y colocar las instrucciones que realizará la función para luego cerrar la llave. La sintaxis se muestra a continuación:

function nombre_funcion([$param1, $param2, ... , $paramn]){
//bloque de instrucciones PHP
}

Como se puede ver, son necesarios los siguientes componentes en la declaración de una función:

- La palabra reservada function que debe utilizarse para indicar al intérprete de PHP que se va a crear una nueva función nombre_funcion indica el nombre con el que se va a identificar la función dentro del script PHP para su posterior llamada.
- $param1, $param2, ... , $paramn representan los parámetros necesarios para que la función pueda ser ejecutada. Los parámetros han de expresarse siempre entre paréntesis y separados por comas. Incluso si la función no necesitara ningún parámetro, deberán utilizarse los paréntesis.
- Bloque de instrucciones PHP, representa el conjunto de sentencias o instrucciones que se van a ejecutar cada vez que se haga una llamada a la función.

Argumentos o parámetros de la función

El diseño de una función puede o debería incluir la aceptación opcional de uno o más argumentos o parámetros que representarán valores que se pasan desde el exterior a la función. Esto permite que la función sea independiente del exterior, haciéndola más portable.

Se puede entender un argumento o parámetro como una variable con ámbito local a la función que sirve para almacenar valores que son pasados a la función.

Los argumentos o parámetros se especifican colocando nombres de variables entre los paréntesis de la definición de la función:

```
function myfunction($parametro1, $parametro2){
//Código de la función
}
```

Devolución de valores

Casi siempre será importante que la función creada devuelva un valor antes de finalizar la ejecución. Aunque también existirán casos en los que la función pudiera realizar una tarea sin que devuelve valor alguno. La palabra reservada utilizada para devolver valores en las funciones de PHP es return, igual que en otros lenguajes que usted ya conoce. Veamos los siguientes ejemplos:

```
function square ($num) {
return $num * $num;
```

```
}
echo square(4); // obtiene '16'.
----------------------------------------
function residuo($dividendo, $divisor){
$res = $dividendo % $divisor;
return $res;
}
```

No se puede devolver múltiples valores desde una función, pero un efecto similar se puede conseguir devolviendo una lista.

```
function small_numbers(){
return array(0, 1, 2);
}
list($zero, $one, $two) = small_numbers();
```

Tipos de funciones.

En PHP podemos utilizar las funciones incorporadas del lenguaje (abs(), list(), array(), chr(), intval(), trim(), substr(), printf(), time(), date(), etc.) o pueden realizarse funciones definidas por el usuario.

Las funciones incorporadas del lenguaje se comportan de la forma predeterminada para la que fueron creadas.

Existen muchas funciones incorporadas o predefinidas en PHP. Algunas ya las hemos utilizado en guías anteriores. Por ejemplo: gettype(), settype(), print(), printf(), substr(), date(), etc.

Las funciones definidas por el usuario, o personalizadas, son creadas por el programador y su funcionalidad y utilidad dependen en gran medida de la habilidad del programador. La ventaja de las segundas es que el programador posee un control completo sobre ellas, ya que puede hacer que una función se comporte exactamente del modo que se desea.

Parámetros de las funciones

La información puede suministrarse a las funciones mediante la lista de parámetros, una lista de variables y/o constantes separadas por comas.

PHP soporta pasar parámetros por valor (el comportamiento por defecto), por referencia, y parámetros por defecto. Listas de longitud variable de parámetros sólo están soportadas en PHP4 y posteriores. Un efecto similar puede conseguirse en PHP3 pasando un array de parámetros a la función:

```
function takes_array($input) {
echo "$input[0] + $input[1] = ", $input[0]+$input[1];
}
```

Pasar parámetros por referencia

Por defecto, los parámetros de una función se pasan por valor (de manera que si se cambia el valor del argumento dentro de la función, el valor pasado fuera de ella no se ve alterado). Si se desea permitir a una función modificar el valor de sus parámetros, deben ser pasados por referencia.

Para hacer que una función pase sus parámetros por referencia existen dos métodos:

Anteponiendo un símbolo de ampersand (&) al nombre del parámetro o argumento en la definición de la función. Esto hará que siempre que se mande un parámetro a la función en la llamada, este será enviado por referencia. Por ejemplo:

```
function add_some_extra(&$string){
$string .= ' y algo más.';
}
$str = 'Esto es una cadena, ';
add_some_extra($str);
echo $str; // Saca 'Esto es una cadena, y algo más.'
```

Anteponer el símbolo de ampersand (&) al nombre del parámetro en la llamada a la función. Este método se debe aplicar si se desea pasar una variable por referencia a una función que no toma el parámetro por referencia por defecto; es decir, si no se especificó en la definición de la función.

```
$num = 10;
function called_function($number){ //En la declaración no indica que
//el parámetro sea por referencia

$number = $number +1;
echo $number, "\n";
}

called_function(&$num); //Se le pasa el parámetro por
//referencia a la función
echo $num, "\n";
```

Parámetros por defecto

Una función puede definir valores por defecto para los parámetros escalares estilo C++. Esto significa que si al llamar a una función, que requiere parámetros, estos no se especifican, se le asignará a estos un valor por defecto. Veamos el siguiente ejemplo:

```
function makecoffee($type = "cappucino"){
return "Hacer una taza de $type.\n";
}
echo makecoffee(); //Se llama a la función sin indicar parámetro
echo makecoffee("espresso"); //Se llama a la función usando parámetro
```

La salida del fragmento anterior es:
Hacer una taza de cappucino.
Hacer una taza de expresso.

Funciones variables

En la mayoría de casos las llamadas a función se realizan con el identificador de la función de forma directa, sin embargo, en PHP es posible asignar el identificador a una variable de cadena y hacer que en la llamada a función se utilice esta variable en lugar del nombre o identificador de la función. Por ejemplo:

```
$funcion = "potencia";
```

```
$valor = 3;
echo "$valor elevado al cuadrado es: " . $funcion(3) . "<br />";
//Definición de la función sqrt
function sqrt($numero){
return $numero * $numero;
}
```

III. MATERIALES Y EQUIPO

Para la realización de la guía de práctica se requerirá lo siguiente:

No.	Material	Cantidad
1	Guía de práctica #5: Arreglos	1
2	Computadora con WampServer instalado y funcionando correctamente	1
3	Editor PHP sublime Text o Eclipse PHP	1
4	Memoria USB o disco flexible	1

IV. PROCEDIMIENTO

Indicaciones: Asegúrese de digitar el código de los siguientes ejemplos que se presentan a continuación. Tenga en cuenta que el PHP Designer no es compilador solamente un editor. Por lo tanto, los errores de sintaxis los podrá observar únicamente hasta que se ejecute el script cargando la página en el navegador de su preferencia.

Ejemplo 1: En este ejemplo se ilustra cómo utilizar la característica de funciones variables que se pueden emplear en PHP. Se tiene una matriz con distintos valores de monedas que se desean convertir a cinco tipos de monedas diferentes.
Archivo 1: monedas.php

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8" />
    <title>Convertir entre monedas</title>
    <link rel="stylesheet" href="fonts.css">
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min
.css"
integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf
23Q9Ifjh" crossorigin="anonymous">
</head>
```

```
<body>
<header >
<nav class="navbar navbar-dark bg-primary">
<span class="navbar-text"><h1>Equivalencias entre monedas</h1></span>
</nav>
</header>
<?php
  $conversion =
array("aDolares","aQuetzal","aLempira","aCordova","aColon");
  $precios = range(10,1000,100);
  //Funciones para usar como funciones variables
  //y hacer la tabla de llamadas que permitirá
  //invocar a las funciones del array con retrollamadas
  function aDolares($dato){
      return sprintf("%02.2f", $dato*0.11425);
  }
  function aQuetzal($dato){
      return sprintf("%02.2f", $dato*0.87214);
  }
  function aLempira($dato){
      return sprintf("%02.2f", $dato*2.14132);
  }
  function aCordova($dato){
      return sprintf("%02.2f", $dato*2.60470);
  }
  function aColon($dato){
      return sprintf("%02.2f", $dato*58.1205);
  }
?>
<section>
<article>
<table class="table table-striped">
    <thead>
        <tr>
            <th>Colones (sv)</th>
            <th>Dólares (sv)</th>
            <th>Quetzales (gt)</th>
            <th>Lempiras (ho)</th>
            <th>Córdovas (ni)</th>
            <th>Colones (cr)</th>
        </tr>
    </thead>
    <tbody>
<?php
  for($i=0;$i<sizeof($precios);$i++){
      if($i%2 == 0){
          echo "\t<tr>\n";
      }
      else{
          echo "\t<tr class=\"odd\">\n";
```

```
      }
      echo "\t\t<td>\t\t&cent; " . $precios[$i] . "\t\t</td>\n\t\t";
      for($j=0; $j<sizeof($conversion); $j++):
          $resultado = $conversion[$j];
          switch($resultado):
              case "aDolares":
                  $signo = "$";
                  break;
              case "aQuetzal":
                  $signo = "Q";
                  break;
              case "aLempira":
                 $signo = "L$";
                 break;
              case "aCordova":
                 $signo = "C$";
                 break;
              case "aColon":
                  $signo = "&cent;";
                  break;
          endswitch;
          echo "<td>\t\t$signo " .
number_format($resultado($precios[$i]),3,".",",") . "\t\t</td>\n\t";
    endfor;
    echo "\t</tr>\n";
  }
?>
</table>
</article>
</section>
</body>
</html>
```

Vista en pantalla:

Equivalencias entre monedas

Colones (sv)	Dólares (sv)	Quetzales (gt)	Lempiras (ho)	Córdovas (ni)	Colones (cr)
¢ 10	$ 1.140	Q 8.720	L$ 21.410	C$ 26.050	¢ 581.210
¢ 110	$ 12.570	Q 95.940	L$ 235.550	C$ 286.520	¢ 6,393.260
¢ 210	$ 23.990	Q 183.150	L$ 449.680	C$ 546.990	¢ 12,205.310
¢ 310	$ 35.420	Q 270.360	L$ 663.810	C$ 807.460	¢ 18,017.350
¢ 410	$ 46.840	Q 357.580	L$ 877.940	C$ 1,067.930	¢ 23,829.400
¢ 510	$ 58.270	Q 444.790	L$ 1,092.070	C$ 1,328.400	¢ 29,641.450
¢ 610	$ 69.690	Q 532.010	L$ 1,306.210	C$ 1,588.870	¢ 35,453.500
¢ 710	$ 81.120	Q 619.220	L$ 1,520.340	C$ 1,849.340	¢ 41,265.560
¢ 810	$ 92.540	Q 706.430	L$ 1,734.470	C$ 2,109.810	¢ 47,077.610
¢ 910	$ 103.970	Q 793.650	L$ 1,948.600	C$ 2,370.280	¢ 52,889.650

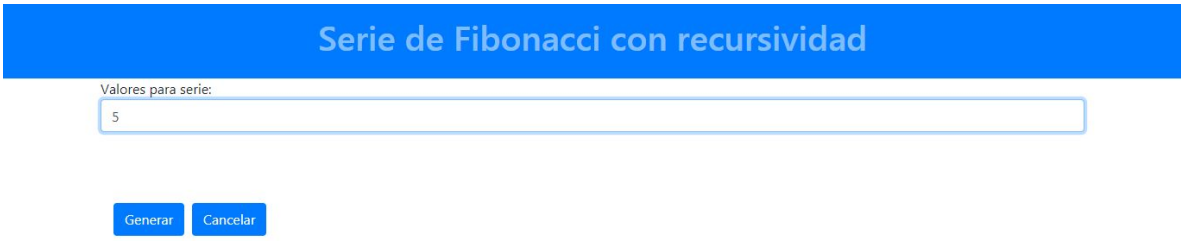
Ejemplo 2: El siguiente ejemplo muestra cómo generar la serie de Fibonacci solicitando al usuario el número de términos de la serie generada. Este dato es validado en el cliente y en el servidor.
Archivo 1: fibonacci.php

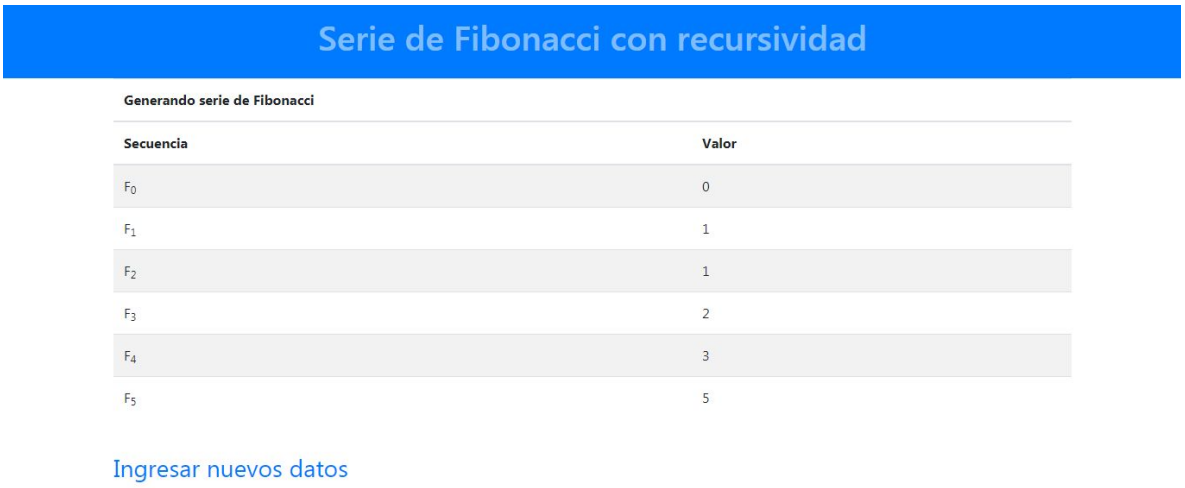
```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8" />
    <title>Serie de Fibonacci</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min
.css"
integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf
23Q9Ifjh" crossorigin="anonymous">
</head>
<body>
<header >
<nav class="navbar navbar-dark bg-primary ">
<span class="navbar-text mx-auto"><h1>Serie de Fibonacci con
recursividad</h1></span>
</nav>
</header>
<div class="container">
<?php
  //Función de Fibonacci recursiva
```

```php
  function fibonacci($n){
      if(($n == 0) || ($n == 1)) return $n;
      return fibonacci($n - 2) + fibonacci($n - 1);
  }

  if(isset($_GET['enviar'])):
      //Verificando el correcto ingreso de los datos
      $numero1 = isset($_GET['txtnumero']) ? $_GET['txtnumero'] : -1;
      if($numero1 == -1):
          $backlink = "<a class='d-block text-muted text-small'
href=\"{$_SERVER['PHP_SELF']}\" title=\"Regresar\">";
          $backlink .= "<span class=\"a-btn-text\">Ingresar dato</span>";
          $backlink .= "<span
class=\"a-btn-slide-text\">nuevamente</span>";
          $backlink .= "</a>";
          echo $backlink;
          exit(0);
      endif;
      $tabla  = "<table class='table table-striped'>";
      $tabla .= "<thead>";
      $tabla .= "<tr><th colspan='2' class='mx-auto'>Generando serie de
Fibonacci</th></tr>";
      $tabla .= "<tr><th scope='col'>Secuencia</th>";
      $tabla .= "<th scope='col'>Valor</th>";
      $tabla .= "<thead>";
      $tabla .= "<tbody>";
      $i = 0;
      //Con este contador se verifica que se generen el número
      //de términos en la serie que indicó el usuario en el formulario
      while($i <= $numero1){
          $class = $i%2 == 0 ? "odd" : "even";
          $tabla .= "<tr class=\"$class\"><td>F<sub>$i</sub></td>";
          $tabla .= "<td>" . fibonacci($i) . "</td></tr>";
          $i++;
      }
      $tabla .= "<tbody></table>";
      echo $tabla;
      echo "<br /><a href=\"{$_SERVER['PHP_SELF']}\" title=\"Regresar\"
class=\"a-btn\">";
      echo "<span class='d-block h3 font-weight-normal'>Ingresar nuevos
datos</span>";

  else:
?>

<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="GET" >

<div class="form-group row">
    <label class="form-check-label" for="numero1">Valores para serie:
</label>
```

```
    <input class="form-control" value="0"   type="text"  name="txtnumero"
id="txtnumero" size="4" maxlength="3" pattern="[0-9]{1,3}" required />
</div>
<div class="alert alert-danger" id="numbersOnly"
style="visibility:hidden;">
  <strong>Dato Erroneo!</strong> Ingrese solo números
</div>
    <input type="submit" name="enviar" value="Generar" class="btn
btn-primary" /> 
    <input type="reset" name="limpiar" value="Cancelar" class="btn
btn-primary" />
</form>
<script src="js/validar.js"></script>
<?php endif; ?>
</div>
</body>
<script src="js/modernizr.custom.lis.js"></script>
</html>
```

Vista en pantalla:





Ejemplo 3: El siguiente ejemplo implementa funciones con lista variable de argumentos para obtener el mayor de una lista de números. Los números son proporcionados directamente en el script PHP.

Archivo 1: elmayor.php

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8" />
    <meta name="viewport"
content="width=device-width,user-scalable=no,initial-scale=1.0,maxi
mum-scale=1.0,minimum-scale=1.0" />
    <title>Trabajando con funciones de lista variable de
argumentos</title>
    <link rel="stylesheet"
href="http://fonts.googleapis.com/css?family=Oswald" />
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootst
rap.min.css"
integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGF
N9MuhOf23Q9Ifjh" crossorigin="anonymous">
    <script src="js/modernizr.custom.lis.js"></script>
    <script src="js/prefixfree.min.js"></script>
</head>
<body>

<section id="container">
<header >
<nav class="navbar navbar-dark bg-primary ">
<span class="navbar-text mx-auto"><h1>Mayor de una lista de
números</h1></span>
</nav>
</header>
<div class="container-fluid">
    <div class="row">
    <?php
    function elMayor(){
        $num_args = func_num_args();
            $args = func_get_args();
        $elmayor = $args[0];
            for($i=1; $i<$num_args; $i++){
            $elmayor = ($elmayor > $args[$i]) ? $elmayor :
func_get_arg($i);
            }
            return $elmayor;
    }
    echo "<div class='col-sm-6 py-5 px-lg-5 border bg-success'>El
mayor de 58, 167, 242, 85, 31, 109, -26, 81, 16, 126 es: " .
        "<h2>" . elMayor(58, 167, 242, 85, 31, 109, -26, 81, 16,
126) . "</h2></div>";
```

```
    ?>

    <?php
        echo "<div class='col-sm-6 py-5 px-lg-5 border
bg-warning'>El mayor de 61, 37, 75, 184, 42, -303, 43, 56, -121,
226, 172, 78, 6, 86 es:" .
        "<h2>" . elMayor(61, 37, 75, 184, 42, -303, 43, 56, -121,
226, 172, 78, 6, 86) . "</div>";
    ?>
    </div>
</div>
</section>
</body>
</html>
```

Vista en pantalla:

Ejemplo 4: #6: El siguiente ejemplo ilustra cómo manejar el envío de campos de formulario de tipo select y checkbox cuando en ambos casos se maneja el nombre del campo como una matriz dentro del script PHP.
Se crea una función en script del lado del servidor que procesará los datos seleccionados en ambos tipos de campos, teniendo en cuenta que en el caso de los datos seleccionados en el campo de formulario de tipo select se creará una lista desordenada (ol), mientras que en el caso de los checkbox que utilizan el mismo nombre en todos los campos creados con ciudades, se creará una lista ordenada (ul). En la función se utilizan dos argumentos, el primero que debe ser la matriz con los datos de la lista y el segundo que indica el tipo de lista que se quiere crear donde los únicos valores posibles deberían ser: "ul", "ol" .
Archivo 1: selectfields.html

```
<!DOCTYPE html>
```

```
<html lang="es">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Estilo de checkboxes, radios y multi select</title>
    <link rel="stylesheet"
href="http://fonts.googleapis.com/css?family=Bitter" />
    <link rel="stylesheet"
href="http://ajax.googleapis.com/ajax/libs/jqueryui/1/themes/ui-ligh
tness/jquery-ui.css" />
    <link rel="stylesheet" href="css/jquery.multiselect.css" />
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstr
ap.min.css"
integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN
9MuhOf23Q9Ifjh" crossorigin="anonymous">
</head>
<body>
    <header >
        <nav class="navbar navbar-light bg-warning ">
        <span class="navbar-text mx-auto"><h1>Ciudades y Destinos de
Vacación</h1></span>
        </nav>
        </header>
<div class="container">
<div class="form-style">
<form name="frmdestinos" action="destinos.php" method="POST">
<div class="form-group">
<h1>
    Selección de
    <span>países y ciudades</span>
</h1>
<select class="form-control"  name="location[]" multiple="multiple"
size="5" id="location">
    <option value="Guatemala">Guatemala</option>
    <option value="El Salvador">El Salvador</option>
    <option value="Costa Rica">Costa Rica</option>
    <option value="Honduras">Honduras</option>
    <option value="Panamá">Panamá</option>
    <option value="Nicaragua">Nicaragua</option>
    <option value="Belice">Belice</option>
</select><br />
</div>
<div class="form-group">
<div id="places " class="form-check">
    <input class="form-check-input" type="checkbox"  name="place[]"
id="place1" value="Ciudad de Guatemala" />
    <label for="place1"><span></span>Ciudad de Guatemala</label><br
/>
```

```
    <input class="form-check-input" type="checkbox" name="place[]"
id="place2" value="Antigua Guatemala" />
    <label for="place2"><span></span>Antigua Guatemala</label><br />
    <input class="form-check-input" type="checkbox" name="place[]"
id="place3" value="Panajachel" />
    <label for="place3"><span></span>Panajachel</label><br />
    <input class="form-check-input" type="checkbox" name="place[]"
id="place4" value="San Salvador" checked="checked" />
    <label for="place4"><span></span>San Salvador</label><br />
    <input class="form-check-input"  type="checkbox" name="place[]"
id="place5" value="Santa Ana" />
    <label for="place5"><span></span>Santa Ana</label><br />
    <input class="form-check-input" type="checkbox" name="place[]"
id="place6" value="Santa Miguel" />
    <label for="place6"><span></span>Santa Miguel</label><br />
    <input class="form-check-input"  type="checkbox" name="place[]"
id="place7" value="San José" />
    <label for="place7"><span></span>San José</label><br />
    <input class="form-check-input"  type="checkbox" name="place[]"
id="place8" value="Puntarenas" />
    <label for="place8"><span></span>Puntarenas</label><br />
    <input class="form-check-input"  type="checkbox" name="place[]"
id="place9" value="San Pedro Sula" />
    <label for="place9"><span></span>San Pedro Sula</label><br />
    <input class="form-check-input"  type="checkbox" name="place[]"
id="place10" value="Tegucigalpa" />
    <label for="place10"><span></span>Tegucigalpa</label><br />
    <input class="form-check-input"  type="checkbox" name="place[]"
id="place11" value="Granada" />
    <label for="place11"><span></span>Granada</label><br />
    <input class="form-check-input" type="checkbox" name="place[]"
id="place12" value="Ciudad de Panamá" />
    <label for="place12"><span></span>Ciudad de Panamá</label><br />
    <input class="form-check-input" type="checkbox" name="place[]"
id="place13" value="Colón" />
    <label for="place13"><span></span>Colón</label><br />
    <input class="form-check-input" type="checkbox" name="place[]"
id="place14" value="Colón" />
    <label for="place14"><span></span>Belmopán</label><br />
</div>
<input type="submit" name="submit" id="enviar" value="Enviar"
class="button btn-primary" />
<input type="reset" name="cancel" value="Cancelar" class="button
btn-primary" />
</div>
</form>
</div>
</div>
</body>
<script src="js/checkFields.js"></script>
```

```
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1/jquery.js"></scri
pt>
<script
src="http://ajax.googleapis.com/ajax/libs/jqueryui/1/jquery-ui.min.j
s"></script>
<script src="js/jquery.multiselect.js"></script>
<script>
    $(function(){
        $("#location").multiselect();
    });
</script>
</html>
```

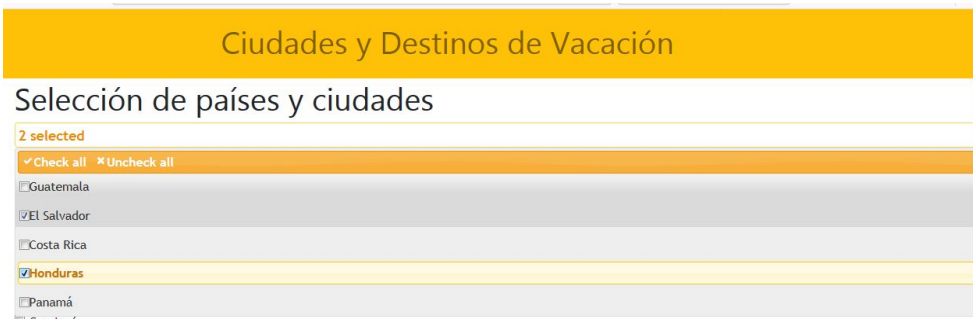
Archivo 2: destinos.php

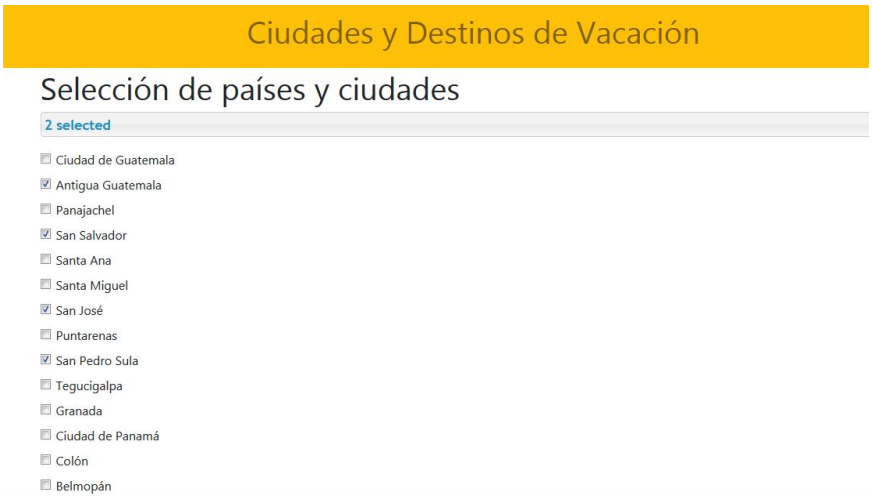
```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Ciudades de destino</title>
    <link rel="stylesheet"
href="http://fonts.googleapis.com/css?family=Bitter" />
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstr
ap.min.css"
integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN
9MuhOf23Q9Ifjh" crossorigin="anonymous">
    <link rel="stylesheet" href="css/fields.css" />
</head>
<body>
<header >
<nav class="navbar navbar-light bg-warning ">
<span class="navbar-text mx-auto"><h1>Ciudades de
Destinos</h1></span>
</nav>
</header>
<section>
<div class="container">
<?php
  //La función espera una matriz con una lista ($lista)
  //ya sea de países o ciudades y un parámetro opcional
  //con el tipo de lista que tendrá un valor por defecto "ul"
  function createList($lista, $tipo="ul"){
      //Inicializando variables para ambos tipos de listas HTML
      $ullist = "";
      $ollist = "";
```

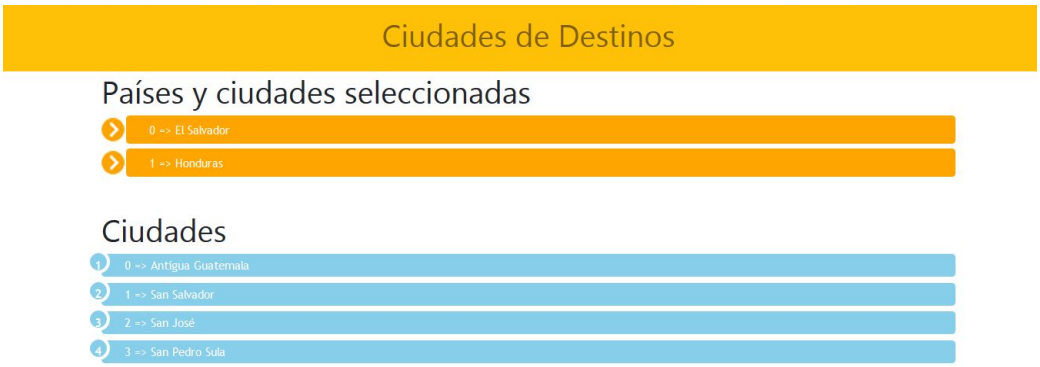
```
      switch($tipo):
          case "ul":
              $ullist .= "<article id=\"countries\">\n";
              $ullist .= "\t<h1>\n";
              $ullist .= "\t\tPaíses y ciudades\n";
              $ullist .= "\t\t<span>seleccionadas</span>\n";
              $ullist .= "\t</h1>\n";
              $ullist .= "\t<ul class=\"imglist\">\n";
              foreach($lista as $key => $value):
                  $ullist .= "\t\t<li><a
href=\"javascript:void(0)\">$key => $value</a></li>\n";
              endforeach;
              $ullist .= "\t</ul>\n";
              $ullist .= "</article>\n";
              print $ullist;
              break;
          case "ol":
              $ollist .= "<article id=\"cities\">\n";
              $ollist .= "\t<h1><span>Ciudades</span></h1>\n";
              $ollist .= "\t<div class=\"numberlist\">\n";
              $ollist .= "\t\t<ol>\n";
              foreach($lista as $key => $value):
                  $ollist .= "\t\t\t<li><a
href=\"javascript:void(0)\">$key => $value</a></li>\n";
              endforeach;
              $ollist .= "\t\t</ol>\n";
              $ollist .= "\t</div>\n";
              $ollist .= "</article>";
              print $ollist;
              break;
          default:
              print "<p>No está definido este tipo de lista</p>";
              break;
      endswitch;
  }

  //Inicia el procesamiento del formulario
  if(isset($_POST['submit'])):
      //Análisis de los elementos de campo select
      if(is_array($_POST['location'])):
          //Si se tiene una matriz invocar a la función
          //createList para crear la lista UL
          createList($_POST['location']);
      else:
          echo "Se esperaba una lista, no un valor escalar.";
          exit(0);
      endif;

      //Análisis de los elementos checkbox
      extract($_POST);
```

```
      if(is_array($place)):
          createList($place, "ol");
      endif;
  else:
      print "<p>El procesamiento del formulario requiere que se haya
presionado el botón Enviar.</p>";
      print "<a href=\"selectfields.html\">Regresar</a>";
      exit(0);
  endif;
?>
</div>
</section>
</body>
</html>
```

Vista en pantalla:







V. DISCUSIÓN DE RESULTADOS

1. Realice un script PHP que utilice una función para que dada una lista de números enteros ingresada por el usuario (como mínimo 2 números) permita encontrar el número mayor y menor de dicha lista utilizando dos funciones con lista variable de argumentos, una para encontrar el número mayor y otra para encontrar el número menor. Los números deben ser ingresados mediante un campo input de tipo número (type="number") que mediante un botón agregar se añadirán en un campo select de múltiple selección. Cuando se hayan ingresado varios número, el usuario debe seleccionar todos o una cantidad de valores ingresados que no debe ser menor que dos. Si se intenta enviar uno solo o ninguno de los números debe lanzarse un mensaje de alerta con JavaScript. La solución debe permitir visualizar la lista de números ingresados e indicar cuáles han sido el número mayor y el número menor encontrados por sus funciones.
2. Realice un script PHP que genere una tabla de multiplicar solicitando el número de la tabla al usuario a través de un formulario, luego, haciendo uso de una función se obtenga el valor ingresado y se cree la tabla de multiplicar de ese número. Realizar una función que no recibe argumentos ni retorna valor alguno. Se debe validar el valor numérico que el usuario ingresa haciendo uso de validación en el cliente con JavaScript.

VI. INVESTIGACIÓN COMPLEMENTARIA

1. Investigue cómo construir una aplicación modular creando un sitio web en partes, luego realizar un ejemplo para armar la página principal de un sitio web. Utilizando esta técnica de construcción, donde se utilizan archivos .inc.php

VII. BIBLIOGRAFIA

- Gutiérrez, Abraham / Bravo, Ginés. PHP 5 a través de ejemplos. Editorial Alfaomega RAMA. 1ra edición. México. Junio 2005.
- Gil Rubio, Francisco Javier/Villaverde, Santiago Alonso/Tejedor Cerbel, Jorge A. Creación de sitios web con PHP 5. Editorial McGraw-Hill. 1ra edición. Madrid, España, 2006.

- John Coggeshall. La Biblia de PHP 5. 1ra Edición. Editorial Anaya Multimedia. Madrid España.
- http://www.php.net/manual/en