

| | | | | | | | | | | | | | | | | | | | |
|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
| | | | | | | | | | | | | | | | | | | | |
|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|



Internal Assessment Test 1– MAY. 2023

| | | | | | | | | | |
|--------------|----------------------------------|------------------|-----------------|-------------------|-----------|-------------|-----------|------------------|----------------|
| Sub: | Advanced Web Technologies | | | | | | | Sub Code: | 20MCA41 |
| Date: | 20/5/2023 | Duration: | 90 min's | Max Marks: | 50 | Sem: | IV | Branch: | MCA |

Note : Answer FIVE FULL Questions, choosing ONE full question from each Module

PART I

1 Explain the various selection statements in PHP, write a PHP code to explain switch statement.

OR

2 Explain session tracking in PHP with programming example

PART II

3 Explain cookies in PHP with programming example

OR

4 Explain various scalar types and operator available in PHP

PART III

5 Illustrate steps to join Web form with database in PHP

| MARKS | OBE | |
|-------|-----|-----|
| | CO | RBT |
| [10] | CO1 | L2 |
| [10] | CO1 | L3 |
| [10] | CO1 | L3 |
| [10] | CO1 | L2 |
| [10] | CO1 | L3 |

OR

6 Explain with examples numeric pre-defined functions in PHP

PART IV

7 Write a PHP program to insert name and age information entered by the user into a table created using MySQL and to display the current contents of this table..

OR

8 Write a PHP program to read student data from an XML file and store into the MySQL database. Retrieve and display

PARTV

9 How to create an array in PHP? Explain different functions for dealing with array?

OR

10 A file contains lines of employee data, where each line has name:age:department code:salary. Write a PHP program to generate the following output:
i)The names of all the employee whose names end with “son”
ii)Percentage of employee under 40 years old
iii)Average salary of employees under 40 years old
iv) An alphabetical list of employees who are under 40 years old and who have salaries more than \$40,000

| | | |
|------|-----|----|
| [10] | CO1 | L3 |
| [10] | CO1 | L4 |
| [10] | CO1 | L4 |
| [10] | CO1 | L3 |
| [10] | CO1 | L4 |

Q1) Explain the various selection statements in PHP, write a PHP code to explain switch statement.

Selection statements

The selection statements in PHP allows the PHP processor to select a set of statements based on the truth value of a condition or Boolean expression. Selection statements in PHP are if, if-else, elseif ladder and switch statement.

Syntax of *if* is given below:

```
if(condition / expression)
{
    statements(s);
}
}
```

Syntax of *if-else* is given below:

```
if(condition / expression)
{
    statements(s);
}
else
{
    statements(s);
}
```

Syntax of *elseif* ladder is given below:

```
if(condition / expression)
{
    statements(s);
}
elseif(condition / expression)
{
    statements(s);
}
elseif(condition / expression)
{
    statements(s);
}
else
{
    statements(s);
}
```

Syntax of *switch* statement is shown below:

```
switch(expression)
{
    case label1:
        statement(s);
        break;
    case label2:
        statement(s);
        break;
    case label3:
        statement(s);
        break;
    default:
        statement(s);
}
```

The labels for *case* statements can be either an integer, double or a string. The *default* block is optional. If the *break* statement is absent, the following cases will also execute until a *break* statement is found.

```
switch ($borderize) {
    case "0": print "<table>";
                break;
    case "1": print "<table border = '1'>";
                break;
    case "4": print "<table border = '4'>";
                break;
    case "8": print "<table border = '8'>";
                break;
    default: print "Error-invalid value: $borderize <br />";
}
```

Q2) Explain session tracking in PHP with programming example

In many cases, information about a session is needed only during the session. Also, the needed information about a client is nothing more than a unique identifier for the session, which is commonly used in shopping cart applications. For these cases, a different process, named session tracking, can be used.

- Rather than using one or more cookies, a single session array can be used to store information about the previous requests of a client during a session.
- In particular, session arrays often store a unique session ID for a session.
- One significant way that session arrays differ from cookies is that they can be stored on the server, whereas cookies are stored on the client.
- In PHP, a session ID is an internal value that identifies a session. Session IDs need not be known or handled in any way by PHP scripts.
- PHP is made aware that a script is interested in session tracking by calling the session start function, which takes no parameters. The first call to session start in a session causes a session ID to be created and recorded.

- On subsequent calls to session_start in the same session, the function retrieves the \$_SESSION array, which stores any session variables and their values that were registered in previously executed scripts in this session.
- Session key/value pairs are created or changed by assignments to the \$ SESSION array.
- They can be destroyed with the unset operator.
- Consider the following example: session_start();
if (!isset(\$_SESSION["page_number"]))
\$_SESSION["page_number"] = 1;
\$page_num = \$_SESSION["page_number"];
print("You have now visited \$page_num page(s)
");
\$_SESSION["page_number"]++;
- If this is not the first document visited that calls session start and sets the page_number session variable, this script will produce the specified line with the last set value of \$_SESSION ["page_number") .
- If no document that was previously visited in this session set page_number, this script sets page_number to 1, produces the line 'You have now visited 1 page(s)', and increments page_number

Program

```

<html>
<head>
<title>Page Views </title>
</head>
<body bgcolor="#cCFFCC" text="#003300">
<h1> Web Programming Lab</h1>
<p> Welcome to Web Programming Lab </p>
<hr />
<p style="color:blue; font-style: italic">
<?php
session_start();
//session_register("count");
$_SESSION["count"];
if(!isset($_SESSION["count"]))
{
$_SESSION["count"] = 0;
echo "Counter initialized... <br />";
}
else { $_SESSION["count"]++; }
echo "Number of Page Views : <b>$_SESSION[count]</b></p>";
?>
<p>Reload this page to increment</p>
</body>
</html>

```

Q3) Explain cookies in PHP with programming example

- A cookie is a small object of information that consists of a name and a textual value. A cookie is created by some software system on the server.
- The header part of an HTTP communication can include cookies. So, every request sent from a browser to a server, and every response from a server to a browser, can include one or more cookies.

- At the time it is created, a cookie is assigned a lifetime. When the time a cookie has existed reaches its associated lifetime, the cookie is deleted from the browser's host machine.
- Cookie is set in PHP with setcookie function
- First parameter is cookie's name given as a string. The second, if present, is the new value for the cookie, also a string. If the value is absent, setcookie undefines the cookie.
- The third parameter, when present, is the expiration time in seconds for the cookie, given as an integer.
- The default value for the expiration time is zero, which specifies that the cookie is destroyed at the end of the current session. When specified, the expiration time is often given as the number of seconds in the UNIX epoch, which began on January 1, 1970. The time function returns the current time in seconds. So, the cookie expiration time is given as the value returned from time plus some number.
- For example,
setcookie("voted", "true", time() + 86400);

This call creates a cookie named "voted" whose value is "true" and whose lifetime is one day (86,400 is the number of seconds in a day).

- To delete a cookie, use the setcookie() function with an expiration date in the past:

```
setcookie("voted", "true", time() - 86400);
```

- All cookies that arrive with a request are placed in the implicit \$ COOKIES array, which has the cookie names as keys and the cookie values as values.
- We can retrieve the value of the cookie using the global variable \$_COOKIE

```
$_COOKIE[$cookie_name]  
Eg. $_COOKIE["voted"]
```

Program

```
<html>  
<head>  
<title>Last Visit using Cookies</title>  
</head>  
<body bgcolor="#cCCFFCC" text="#003300">  
<h1> Web Programming Lab</h1>  
<p> Welcome to Web Programming Lab </p>  
<hr />  
<p style="color:blue; font-style: italic">  
<?php  
date_default_timezone_set('Asia/Calcutta');  
//Calculate 60 days in the future  
//seconds * minutes * hours * days + current time  
// set expiry date to two months from now  
$inTwoMonths = 60 * 60 * 24 * 60 + time();  
setcookie('lastVisit', date("G:i - m/d/y"), $inTwoMonths);  
if(isset($_COOKIE['lastVisit']))  
{  
$visit = $_COOKIE['lastVisit'];  
echo "Last Visited on : ".$visit;  
}  
else
```

```
echo "You've got some old cookies!";
?>
</p>
</body>
</html>
```

Q4) Explain various scalar types and operator available in PHP

PHP has four scalar types, Boolean, integer, double and string.

Integer type

PHP has a single integer type, named `integer`. This type corresponds to the `long` type of C and its successors, which means its size is that of the word size of the machine on which the program is run. In most cases, this is 32 bits, or a bit less (not fewer) than ten decimal digits.

Double Type

PHP's double type corresponds to the `double` type of C and its successors. Double literals can include a decimal point, an exponent, or both. The exponent has the usual form of an `E` or an `e`, followed by a possibly signed integer literal. There does not need to be any digits before or after the decimal point, so both `.345` and `345.` are legal double literals.

String Type

String literals are defined with either single (`'`) or double quotes (`"`) delimiters. In single-quoted string literals, escape sequences, such as `\n`, are not recognized as anything special, and the values of embedded variables are not substituted. (This substitution is called *interpolation*.) In double-quoted string literals, escape sequences are recognized, and embedded variables are replaced by their current values. For example, the value of

```
'The sum is: $sum'
```

is exactly as it is typed. However, assuming the current value of `$sum` is `10.2`, the value of

```
"The sum is: $sum"
```

is

```
The sum is: 10.2
```

If a double-quoted string literal includes a variable name but you do not want it interpolated, precede the first character of the name (the dollar sign) with a backslash (`\`). If the name of a variable that is not set to a value is embedded in a double-quoted string literal, the name is replaced by the empty string.

Double-quoted strings can include embedded newline characters that are created by the `Enter` key. Such characters are exactly like those that result from typing `\n` in the string.

The length of a string is limited only by the available memory on the computer.

Boolean Type

The only two possible values for the Boolean type are `TRUE` and `FALSE`, both of which are case insensitive. Although Boolean is a data type in the same sense as integer, expressions of other types can be used in Boolean context. If you use a non-Boolean expression in Boolean context, you obviously must know how it will be interpreted.

If an integer expression is used in Boolean context, it evaluates to `FALSE` if it is zero; otherwise, it is `TRUE`.

If a string expression is used in Boolean context, it evaluates to `FALSE` if it is either the empty string or the string `"0"`; otherwise, it is `TRUE`. This implies that the string `"0.0"` evaluates to `TRUE`.

The only double value that is interpreted as `FALSE` is exactly `0.0`. Because of rounding errors, as well as the fact that the string `"0.0"` evaluates to `TRUE`, it is not a good idea to use expressions of type double in Boolean context. A value can be very close to zero, but because it is not exactly zero, it will evaluate to `TRUE`.

Operators are used in expressions to perform operations on operands. There are several operators supported by PHP which are categorized into following categories:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Increment/Decrement operators
- Logical operators
- String operators
- Array operators

Arithmetic Operators

PHP arithmetic operators are used along with numbers to perform operations like addition, subtraction, multiplication etc. Below is a list of arithmetic operators:

| Operator | Name | Example | Description |
|----------|----------------|--------------|--------------------------------------|
| + | Addition | $\$x + \y | Sum of x and y |
| - | Subtraction | $\$x - \y | Difference of x and y |
| * | Multiplication | $\$x * \y | Product of x and y |
| / | Division | $\$x / \y | Quotient of x divided by y |
| % | Modulus | $\$x \% \y | Remainder of x divided by y |
| ** | Exponentiation | $\$x ** \y | Result of x raised to the power of y |

Assignment Operators

PHP assignment operators are used in assignment expressions to store the value of expression in to a variable. Below is a list of assignment operators:

| Assignment | Same as | Description |
|------------|--------------|--|
| $x = y$ | $x = y$ | Assigning value of y to x |
| $x += y$ | $x = x + y$ | Adding x and y and store the result in x |
| $x -= y$ | $x = x - y$ | Subtracting y from x and store the result in x |
| $x *= y$ | $x = x * y$ | Multiplying x and y and store the result in x |
| $x /= y$ | $x = x / y$ | Dividing x by y and store the quotient in x |
| $x \% = y$ | $x = x \% y$ | Dividing x by y and store the remainder in x |

Increment/Decrement Operators

The increment/decrement operators are used to increment the value of variable by 1 or decrement the value of variable by 1. The increment operator is `++` and decrement operator is `--`.

Relational or Comparison Operators

PHP comparison operators are used to compare two values and are frequently seen in Boolean expressions. Below is a list of comparison operators:

| Operator | Name | Example | Description |
|----------|--------------------------|-------------|--|
| == | Equal | \$x == \$y | Returns true if x and y are equal |
| === | Identical | \$x === \$y | Returns true if x and y are equal and of same type |
| != | Not equal | \$x != \$y | Returns true if x and y are not equal |
| !== | Not identical | \$x !== \$y | Returns true if x and y are not equal and of same type |
| < | Less than | \$x < \$y | Returns true if x is less than y |
| <= | Less than or equal to | \$x <= \$y | Returns true if x is less than or equal to y |
| > | Greater than | \$x > \$y | Returns true if x is greater than y |
| >= | Greater than or equal to | \$x >= \$y | Returns true if x is greater than or equal to y |
| <> | Not equal | \$x <> \$y | Returns true if x and y are not equal |

Logical Operators

PHP logical operators are used find the Boolean value of multiple conditional expressions. Below is a list of logical operators:

| Operator | Name | Example | Description |
|----------|------|-------------|--|
| and | And | \$x and \$y | Returns true when both x and y are true |
| or | Or | \$x or \$y | Returns true when either x or y or both of them are true |
| xor | Xor | \$x xor \$y | Returns true when either x or y is true |
| && | And | \$x && \$y | Returns true when both x and y are true |
| | Or | \$x \$y | Returns true when either x or y or both of them are true |
| ! | Not | !\$x | Returns true when x is false and vice versa |

String Operators

PHP provides two operators which are used with strings only. They are listed below:

| Operator | Name | Example | Description |
|----------|--------------------------|-----------------|--------------------------------|
| . | Concatenation | \$str1.\$str2 | str1 and str2 are concatenated |
| .= | Concatenation Assignment | \$str1.= \$str2 | str2 is appended to str1 |

Array Operators

Below is a list of operators which are used with arrays:

| Operator | Name | Example | Description |
|----------|----------|------------|---|
| == | Equality | \$x== \$y | Returns true if x and y have the same key-value pairs |
| === | Identity | \$x=== \$y | Returns true if x and y have the same key-value |

| | | | |
|-----|--------------|------------|---|
| | | | pairs in same order and are of same type |
| != | Inequality | \$x!= \$y | Returns true if x and y are not equal |
| !== | Non-Identity | \$x!== \$y | Returns true if x and y are not identical |
| <> | Inequality | \$x<> \$y | Returns true if x and y are not equal |
| + | Union | \$x+\$y | Returns union of x and y |

Q5) Illustrate steps to join Web form with database in PHP

13.6.2 Connecting to MySQL and Selecting a Database

The PHP function `mysql_connect` connects a script to a MySQL server. This function takes three parameters, all of which are optional. The first is the host that is running MySQL; the default is `localhost` (the machine on which the script is running). The second parameter is the username for MySQL; the default is the username in which the PHP process runs. The third parameter is the password for the database; the default is blank (works if the database does not require a password). For example, if the default parameters were acceptable, we could use the following:

```
$db = mysql_connect();
```

Of course, the connect operation could fail, in which case the value returned would be `false` (rather than a reference to the database). Therefore, the call to `mysql_connect` usually is used in conjunction with `die`.

The connection to a database is terminated with the `mysql_close` function. This function is not necessary when using MySQL through a PHP script because the connection will be closed implicitly when the script terminates.

When running MySQL from the command line, a database must be selected as the current, or focused, database. This is also necessary when using MySQL through PHP; it is accomplished with the `mysql_select_db` function, as shown in the following:

```
mysql_select_db("cars");
```

13.6.3 Requesting MySQL Operations

MySQL operations are requested through the `mysql_query` function. Typically, the operation, in the form of a string literal, is assigned to a variable. Then `mysql_query` is called with the variable as its parameter. For example:

```
$query = "SELECT * from Corvettes";  
$result = mysql_query($query);
```

The return value from `mysql_query` is used to identify, internally, the data that resulted from the operation. In most cases, the first thing to do with the result is to determine the number of rows. This is obtained with the `mysql_num_rows` function, which is given the result value returned by `mysql_query`, as shown in the following:

```
$num_rows = mysql_num_rows($result);
```

The number of fields in a result row can be determined with `mysql_num_fields`, as shown in the following:

```
$num_fields = mysql_num_fields($result);
```

The rows of the result can be retrieved into several different forms. We will use `mysql_fetch_array`, which returns an array of the next row. Then the field values can be obtained by subscripting the return array with the column names. For example, if the result of a query had columns for `State_id` and `State`, we could display the results with the following code:

```
$num_rows = mysql_num_rows($result);  
  
for ($row_num = 1; $row_num <= $num_rows; $row_num++) {  
    $row = mysql_fetch_array($result);  
    print "<p> Result row number" . $row_num .  
        ". State_id: ";  
    print htmlspecialchars($row["State_id"]);  
    print " State: ";  
    print htmlspecialchars($row["State"]);  
    print "</p>";  
}
```

The situation in which the column names are not known is considered in Section 13.6.4, which includes a complete example of accessing a database through PHP and MySQL.

Q6) Explain with examples numeric pre-defined functions in PHP

Table 11.2 Some useful predefined functions

| Function | Parameter Type | Returns |
|----------|---------------------|--|
| floor | Double | Largest integer less than or equal to the parameter |
| ceil | Double | Smallest integer greater than or equal to the parameter |
| round | Double | Nearest integer |
| srand | Integer | Initializes a random number generator with the parameter |
| rand | Two numbers | A pseudorandom number greater than the first parameter and smaller than the second |
| abs | Number | Absolute value of the parameter |
| min | One or more numbers | Smallest |
| max | One or more numbers | Largest |

Example

| Function | Input | Output |
|----------|---------------|-----------|
| floor | floor(20.46) | 20 |
| ceil | ceil(20.46) | 21 |
| round | round(20.46) | 20 |
| round | round(20.46) | 20 |
| rand | rand() | 854377889 |
| rand | rand(10,100) | 40 |
| abs | abs(-15) | 15 |
| min | min(2,4,8,10) | 2 |
| max | max(2,4,8,10) | 10 |

Q7) Write a PHP program to insert name and age information entered by the user into a table created using MySQL and to display the current contents of this table

7.xhtml

```
<html>
<body>
<form action=" 7.php" method="POST">
<table align="center" width="100" height="100">
<caption> Student Information</caption>
<tr><td>Name:</td><td><input type="text" name="name"></td></tr>
<tr><td>Age :</td><td><input type="text" name="age"></td></tr>
<tr><td></td><td><input type="submit" value="Submit"></td></tr></table></form>
</body>
</html>
```

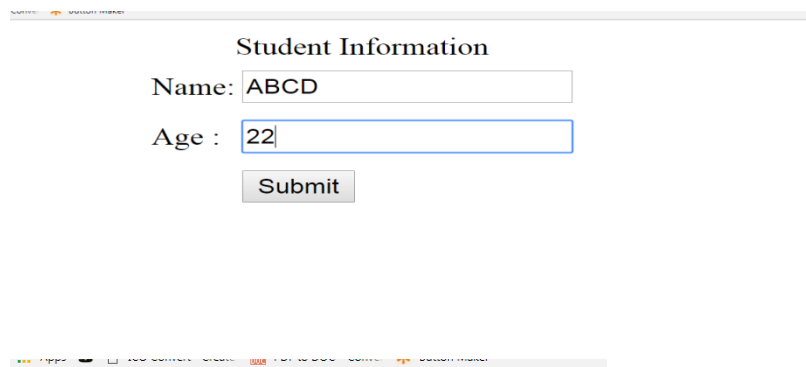
7.php

```
<?php
$name=$_POST['name'];
$age=$_POST['age'];
mysql_connect("localhost","root","");
mysql_select_db("web1");
mysql_query("insert into p3(name,age) values('$name','$age')");

$result=mysql_query("select * from p3 where name='$name' and age='$age'");
$row=mysql_fetch_row($result);

?>
<table>
<tr>
<th>Name</th>
<th>age</th>
</tr>
<tr>
<td><?php echo $row[0];?></td>
<td><?php echo $row[1];?></td>
</tr>
</table>
```

Output



The screenshot shows a web browser window with a form titled "Student Information". The form has two input fields: "Name:" with the value "ABCD" and "Age :" with the value "22". Below the fields is a "Submit" button. The browser's address bar shows a URL starting with "http://localhost".

Name age
ABCD 22

Q8) Write a PHP program to read student data from an XML file and store into the MySQL database. Retrieve and display

8.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<student_info>
<student>
```

```

<usn>1CR17MCA01</usn>
<name>Ajay</name>
</student>
<student>
<usn>1CR17MCA02</usn>
<name>Akshatha</name>
</student>
<student>
<usn>1CR17MCA58</usn>
<name>Piyush</name>
</student>
<student>
<usn>1CR17MCA59</usn>
<name>Taj</name>
</student>
</student_info>

```

8.php

```

<html>
  <body>
    <form name="form1" method="post" action="8.php">
      Enter Name <input type="text" name="stname">
      <input type="submit" name="submit" value="search">
    </form>
  </body>
</html>

```

```

<?php
$con = mysql_connect("localhost","root","");
if (!$con)
{
    die('Could not connect: ' . mysql_error());
}

mysql_select_db("web1", $con);
$lib = simplexml_load_file("7.xml");

$i = "delete from student";
$result = mysql_query($i);
foreach($lib as $stu)
{
    $usn= $stu->usn;
    $name=$stu->name;
    $i="insert into student(usn,name) values('$usn','$name)";
    mysql_query($i);
}

if(($_SERVER["REQUEST_METHOD"]=="POST")||($_SERVER["REQUEST_METHOD"]=="po
st"))
{

```

```

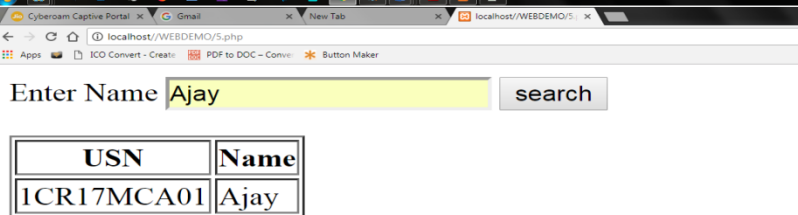
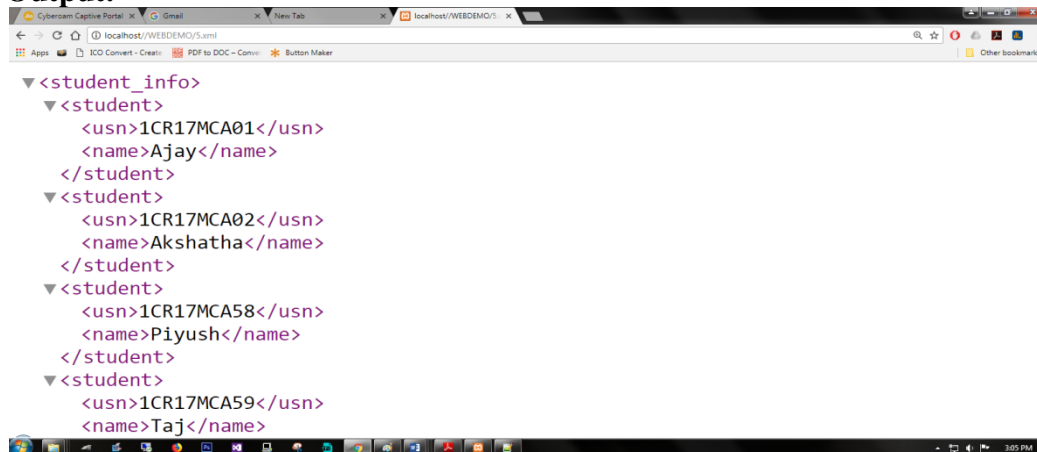
$stname = $_POST["stname"];

$result = mysql_query("SELECT * from student where name LIKE '%" . $stname . "%'");
echo "<table border='1'><tr><th>USN</th><th>Name</th></tr>";

while($row = mysql_fetch_array($result))
{
    echo "<tr><td>" . $row['usn'] . "</td><td>" . $row['name'] . "</td></tr>";
}
echo "</table>";
}
?>

```

Output:-



Q9) How to create an array in PHP? Explain different functions for dealing with array?

Arrays

Array is a collection of heterogeneous elements. There are two types of arrays in PHP. First type of array is a normal one that contains integer keys (indexes) which can be found in any typical programming languages. The second type of arrays is an *associative array*, where the keys are strings. Associative arrays are also known as hashes.

Array Creation

Arrays in PHP are dynamic. There is no need to specify the size of an array. A normal array can be created by using the integer index and the assignment operator as shown below:

```
$array1[0] = 10;
```

If no integer index is specified, the index will be set to 1 larger than the previous largest index value used. Consider the following example:

```
$array2[3] = 5;  
$array2[ ] = 90;
```

In the above example, 90 will be stored at index location 4.

There is another way for creating an array, using the *array* construct, which is not a function. The data elements are passed to the *array* construct as shown in the below example:

```
$array3 = array(10, 15, 34, 56);  
$array4 = array( );
```

In the above example *array4* is an empty array which can be used later to store elements.

A traditional array with irregular indexes can be created as shown below:

```
$array5 = array(3 => 15, 4 => 37, 5 => 23);
```

The above code creates an array with indexes 3, 4 and 5.

An associative array which contains named keys (indexes) can be created as shown below:

```
$ages = array("Ken" => 29, "John" => 30, "Steve" => 26, "Bob" => 28);
```

An array in PHP can be a mixture of both traditional and associative arrays.

Array Functions

The array elements can be manipulated or accessed in different ways. PHP has an extensive list of predefined functions that comes in handy while working with arrays. Some these functions are mentioned below:

| Function Name | Parameters | Description |
|------------------|---------------|---|
| count | One array | Returns the number of elements in the array |
| unset | One array | Deletes the element from memory |
| array_keys | One array | Returns the keys (indexes) as an array |
| array_values | One array | Returns the values as an array |
| array_key_exists | Key, array | Returns <i>true</i> if the key is found, <i>false</i> otherwise |
| is_array | One array | Returns <i>true</i> if the argument is an array, <i>false</i> otherwise |
| in_array | Exp, array | Returns <i>true</i> if <i>exp</i> is in the array, <i>false</i> otherwise |
| explode | Delim, string | Returns an array of substrings based on the <i>delim</i> |
| implode | Delim, array | Returns a string joining the array elements with <i>delim</i> |
| sort | One array | Sorts the values in the array and renames the keys to integer values 0, 1, 2, ... |
| asort | One array | Sorts the values in the array preserving the keys |
| ksort | One array | Sorts the keys in the array preserving the values |
| rsort | One array | Reverse of sort (descending order) |
| rasort | One array | Reverse of asort (descending order) |
| rksort | One array | Reverse of ksort (descending order) |

Q10) A file contains lines of employee data, where each line has name:age:department code:salary. Write a PHP program to generate the following output:

- i) The names of all the employee whose names end with "son"
- ii) Percentage of employee under 40 years old
- iii) Average salary of employees under 40 years old
- iv) An alphabetical list of employees who are under 40 years old and who have salaries more than \$40,000

program.php

```
<?php
$file = fopen("employees.txt","r");
$under_40=0;
$salary_sum=0;
$emplist=[];
echo "Names of all employees whose names end with son : <br/>";
while(! feof($file))
{
    $total_employees++;
    $fline=fgets($file);
    list($name,$age,$dept,$salary)=split(':', $fline);

    if(preg_match("/son$/", $name)){
        echo $name."<br/>";
    }

    if($age<40){
        $under_40++;
        $salary_sum += $salary;
        if($salary>40000 ){
            $emplist[$name]=$salary;
        }
    }
}
```

```

}

if($total_employees>0){
    if($sunder_40>0){
        $percent=100*$sunder_40 / $total_employees;
        echo "<br/>Percent of employees under 40 is :".$percent."%<br/>";
        $avg= $salary_sum/$sunder_40;
        echo "<br/>Averege salary of employees under 40 is :".$avg."<br/>";
        if(array_keys($emplist)){
            echo "<br/>Sorted list of employees under 40, with salries >
\\$40000 <br/>";

            ksort($emplist);
            echo "<table><tr><td>Name</td><td>Salary</td></tr>";
            foreach ($emplist as $key=>$value){
                echo
                "<tr><td>".$key."</td><td>\\$".$value."</td></tr>";
            }
            echo"</table>";
        }

    }
    else{
        echo"There are no employees under 40 who earned over
\\$40000 <br/>";
    }
}
else{
    echo"There are no employees under 40 <br/>";
}
}
else{
    echo"There are no employees ";
}
}
fclose($file);
?>

```

Employee.txt

```

Harry:30:HR:43000
Johnson:28:IT:20000
John Milton:50:Sales:60000
Edison:45:IT:50000
Morrison:46:IT:20000

```

Names of all employees whose names end with son :

Johnson

Edison

Morrison

Percent of employees under 40 is :40%

Average salary of employees under 40 is :\$31500

Sorted list of employees under 40, with salaries > \$40000

Name Salary

Harry \$43000