Lab Manual
for
CSE 326 (E-Commerce and Web Programming Lab)

Credit: 1, Contact hour: 2 Hours Per week

Department of Computer Science & Engineering
Varendra University
Rajshahi, Bangladesh

Varendra University

Department of Computer Science and Engineering

CSE 326

E-Commerce and Web Programming Laboratory

Student ID	
Student Name	
Section	
Name of the Program	
Name of the Department	

INDEX

INSTRUCTIONS FOR LABORATORY

Do’s

- ✓ Do wear ID card and follow dress code.
- ✓ Do log off the computers when you finish.
- ✓ Do ask for assistance if you need help.
- ✓ Do keep your voice low when speaking to others in the LAB.
- ✓ Do ask for assistance in downloading any software.
- ✓ Do make suggestions as to how we can improve the LAB.
- ✓ In case of any hardware related problem, ask LAB in charge for solution.
- ✓ If you are the last one leaving the LAB, make sure that the staff in charge of the LAB is
  informed to close the LAB.
- ✓ Be on time to LAB sessions.
- ✓ Do keep the LAB as clean as possible.

Don’ts

- o Do not use mobile phone inside the lab.
- o Don’t do anything that can make the LAB dirty (like eating, throwing waste papers etc).
- o Do not carry any external devices without permission.
- o Don’t move the chairs of the LAB.
- o Don’t interchange any part of one computer with another.
- o Don’t leave the computers of the LAB turned on while leaving the LAB.
- o Do not install or download any software or modify or delete any system files on any lab
  computers.
- o Do not damage, remove, or disconnect any labels, parts, cables, or equipment.
- o Don’t attempt to bypass the computer security system.
- o Do not read or modify other user’s file.
- o If you leave the lab, do not leave your personal belongings unattended. We are not responsible for any theft.

Varendra University

COURSE SYLLABUS

1	Faculty	Faculty of Science & Engineering
2	Department	Department of CSE
3	Program	B.Sc. in Computer Science and Engineering
4	Name of Course	E-Commerce and Web Programming Lab
5	Course Code	CSE 326
6	Trimester and Year	Spring ,2021
7	Pre-requisites	CSE 313
8	Status	Core Course
9	Credit Hours	2
10	Section	A
11	Class Hours	Not Defined
12	Class Location	Not Defined
13	Name (s) of Academic staff / Instructor(s)	Md. Toufikul Islam
14	Contact	toufikul.cse@gmail.com
15	Office	532, Jahangir Sharoni, Talaimari, Rajshahi-6204
16	Counseling Hours	Not Defined
17	Text Book	1. HTML & XHTML: The Complete Reference, Fourth Edition by Thomas A. Powell 2. CSS Cookbook by By Christopher Schmitt 3. CSS: The Definitive Guide, 3rd Edition By Eric Meyer 4. Learning PHP and MySQL By Michele Davis, Jon Phillips 5. PHP Manual 6. PHP and MySql Bible By *Tim Converse*, Joyce Park, Clark Morgan
18	Reference	1. www.stackoverflow.com 2. w3schools.com
19	Equipment & Aids	1. Lab Sheet 2. Text Book (PDF) 3. XAMPP Software 4. Notepad++/Sublime Text Editor

20	Course Description	This course is designed to provide the student with foundational programming knowledge and skills for application development on the Internet. The student will learn about the Web as a development platform through the use of popular representative languages (such as PHP). The student will learn to plan, design, construct, and integrate basic server-side components of modern web applications including databases and scripts.
21	Course Objectives	The course is designed to provide the background of the following topics: 1. Technical review of the Internet and web applications 2. Overview of static and dynamic web pages 3. Overview of server-side and client-side programming 4. Server-side scripting with inline code using procedural programming techniques 5. Programming web forms that connect with web-enabled databases and perform basic CRUD operations 6. Programming web forms that connect with web-enabled databases and perform basic CRUD operations
22	Learning Outcomes	After the successful completion of this course, students will be able to 1. Utilize a variety of basic programming structures (variables, loops, functions etc.) in a popular scripting environment on a web server. 2. Create scripts that dynamically generate web pages containing valid HTML (Hypertext Markup Language) clearly separating structure, presentation, and behaviour in the web browser. 3. Develop scripts that validate form input on the server. 4. Design, program, and implement web pages that interact with web-enabled databases performing simple CRUD (Create, Read, Update, Delete) operations. 5. Identify the different issues faced by web developers such as security, file management, and web standards-compliancy.
23	Teaching Methods	Lecture, Video Demonstration, Problem Solving, Brainstorming, Project Development, Q/A, Project Presentation
24	Assessment Methods	(see table below)

Assessment Types	**Marks**
Attendance	10%
Laboratory Report	10%
Laboratory Viva-voce	10%
Laboratory Project	20%
Laboratory Final Quiz	30%
Laboratory Examination	20%
Total	**100%**

25	Grading Policy	see nested table below
26	Additional Course Policies	see policies below

25 Grading Policy

Letter Grade	Marks %	Grade Point	Letter Grade	Marks %	Grade Point
A+	80-100	4.00	C+ (Plus)	50-54	2.50
A	75-79	3.75	C (Plain)	45-49	2.25
A-	70-74	3.50	D (Plain)	40-44	2.00
B+	65-69	3.25	F (Fail)	<40	0.00
B	60-64	3.00			
B-	55-59	2.75			

26 Additional Course Policies

1. Lab Reports
Report on previous Experiment must be submitted before the beginning of new experiment. A bonus may be obtained if a student submits a neat, clean and complete lab report.

2. Examination
There will be a lab exam at the end of the semester that will be closed book.

3. Unfair means policy
In case of copying/plagiarism in any of the assessments, the students involved will receive zero marks. Zero Tolerance will be shown in this regard. In case of severe offences, actions will be taken as per university rule.

4. Counseling
Students are expected to follow the counseling hours posted. In case of emergency/unavoidable situations, students can e-mail me to make an appointment. Students are regularly advised to check the piazza course page for updates/materials.

5. Policy for Absence in Class/Exam
If a student is absent in the class for anything other than medical reasons, he/she will not receive attendance. If a student misses a class for genuine medical reasons, he/she must apply with the supporting documents (prescription/medical report). He/she will then have to follow the instructions given by the instructor for make- up.

In case of absence in the mid/final exam for medical grounds, the student must also get his/her application forwarded by the head of the department before a make-up exam can be taken.

It is recommended that the students inform the instructor beforehand through mail if they feel that they will miss a class/evaluation due to medical reasons.

Lab (1): HTML Basic, Headings,Paragraph,Text Formatting and Quotations

❖ HTML Basics:

- All HTML documents must start with a document type declaration: **<!DOCTYPE html>**.
- The HTML document itself begins with **<html>** and ends with **</html>**.
- The visible part of the HTML document is between **<body>** and **</body>**.

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>
<h1>My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>
```

[Task1]. HTML Heading using <h1>,<h2>,…<h6> tags

- HTML headings are defined with the **<h1>** to **<h6>** tags.
- <h1> defines the most important heading. <h6> defines the least important heading.

```
<html>
<body>
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
<h4>This is heading 4</h4>
<h5>This is heading 5</h5>
<h6>This is heading 6</h6>
</body>
</html>
```

[Task2]. HTML Page Paragraphs using <p> tag

```
<html>
<body>
<p>Welcome to Web Lab.</p>
<p>Eight Semester 2019.</p>
</body>
</html>
```

[Task3]. HTML <pre> Element

The HTML <pre> element defines preformatted text. The text inside a <pre> element is displayed in a fixed-width font (usually Courier), and it preserves both spaces and line breaks:

```
<html>
<body>
    <p>The pre tag preserves both spaces and line breaks:</p>
    <pre>
        My Bonnie lies over the ocean.
        My Bonnie lies over the sea.
        My Bonnie lies over the ocean.
    </pre>
</body>
</html>
```

[Task4]. HTML Line Breaks tags

```
<!DOCTYPE html>
<html>
 <body>
     <p>This is<br>a paragraph<br>with line breaks</p>
 </body>
</html>
```

[Task5]. HTML <hr> Tag

Use the <hr> tag to draw horizontal line. Try the next code:

```
<html>
<body>
    <h1>HTML</h1>
    <p>HTML is a language for describing web pages.</p>
    <hr>
    <h1>CSS</h1>
    <p>CSS defines ow to display HTML elements.</p>
</body>
</html>
```

[Task6]. HTML Formatting Elements

HTML also defines special **elements** for defining text with a special **meaning**.

HTML uses elements like `<b>` and `<i>` for formatting output, like **bold** or *italic* text.

Formatting elements were designed to display special types of text:

- <b> - Bold text
- <strong> - Important text
- <i> - Italic text
- <em> - Emphasized text
- <mark> - Marked text
- <small> - Small text
- <del> - Deleted text
- <ins> - Inserted text
- <sub> - Subscript text
- <sup> - Superscript text

```
<html>
<body>
   <p>This text is normal.</p>
   <p><b>This text is bold.</b></p>
   <p><i>This text is italic</i></p>
   <p><em>This text is emphasized</em></p>
   <p>This is a <u>parragraph</u>.</p>
   <p>HTML <small>Small</small> Formatting</p>
   <p>HTML <mark>Marked</mark> Formatting</p>
   <p>My favorite color is <del>blue</del> red.</p>
   <p>My favorite <ins>color</ins> is red.</p>
   <p>This is <sub>subscripted</sub> text.</p>
   <p>This is <sup>superscripted</sup> text.</p>
   <p> <b><i>This text is bold and italic</i></b></p>
</body>
<html>
```

[Task7]. HTML Character Entities

Some characters are reserved in HTML. If you use the less than (<) or greater than (>) signs in your text, the browser might mix them with tags. Character entities are used to display reserved characters in **HTML**.

```
&entity_name;

OR

&#entity_number;
```

To display a less than sign (<) we must write: &lt; or &#60;

Non-breaking Space

A common character entity used in HTML is the non-breaking space: &nbsp; or &#160;

A non-breaking space is a space that will not break into a new line. Two words separated by a non- breaking space will stick together (not break into a new line).

Examples:

<p> 10 &nbsp; km/h </p>

If you write 10 spaces in your text, the browser will remove 9 of them. To add real spaces to your text, you can use the &nbsp; character entity.

© entity name = &copy

[Task8]. HTML Quotation and Citation Elements

The HTML **<q>** element defines a short quotation.

Browsers usually insert quotation marks around the **<q>** element.

```
<html>
<body>
   <p>WWF's goal is to: <q>Build a future where people live in harmony with
nature.</q></p>
<body>
<html>
```

The HTML **<blockquote>** element defines a section that is quoted from another source. Browsers usually indent **<blockquote>** elements.

```
<html>
<body>
   <p>Browsers usually indent blockquote elements.</p>
   <blockquote cite="http://www.worldwildlife.org/who/index.html">
      For 50 years, WWF has been protecting the future of nature.
      The world's leading conservation organization,
      WWF works in 100 countries and is supported by
      1.2 million members in the United States and
      close to 5 million globally.
   </blockquote>
</body>
</html>
```

The HTML **<address>** element defines contact information (author/owner) of a document or an article. The **<address>** element is usually displayed in italic. Most browsers will add a line break before and after the element.

```
<html>
<body>
  <p>The HTML address element defines contact information of a document or article.</p>
  <address>
    Visit us at:<br>
    Example.com<br>
    Box 564, Disneyland<br>
    USA
  </address>
</body>
</html>
```

Lab (2): HTML Lists and Images

❖ Displaying Images on a Web Page

In HTML, images (Gif,JPG, PNG) are defined with the **<img>** tag.The **<img>** tag is empty, it contains attributes only, and does not have a closing tag. The src attribute specifies the URL of the image:
<img src="*url*" alt="*some_text*" width="*width*" height="*height*">

[Task1]. Display images from web site

```
<html>
<body>
  <img src="http://www.w3schools.com/images/w3schools_green.jpg">
</body>
</html>
```

[Task2]. Display images from the same folder of the web page

```
<html>
<body>
   <img src="html.gif">
</body>
</html>
```

[Task3]. Display images from another folder of the web page

```
<html>
<body>
   <img src="/images/html.gif">
</body>
</html>
```

[Task4]. Image alt, width and height attributes

- The **alt** attribute provides an alternate text for an image, if the user for some reason cannot view it.
- You can use the **width and height** attributes to specify the width and height of an image in **pixels**.

```
<html>
<body>
   <imgsrc="html5.gif" alt="HTML5  Icon" width="128" height="128">
</body>
</html>
```

❖ Adding Lists to a Web Page

There are three basic types of HTML lists: **ordered** list, **unordered** list and **definition** list.

[Task5]. The ordered list in HTML

The ordered list which is created by the **ol Element** that begins with the **<ol>** tag and ends with a closing **</ol>** tag. **The attributes required: type** attribute and the **start** attribute.

```
<html>
 <body>
   <ol>
       <li>Coffee</li>
       <li>Tea</li>
       <li>Milk</li>
   </ol>
</body>
</html
```

Type Attribute can take one of the next values:

Type	**Description**
type="1"	The list items will be numbered with numbers (default)
type="A"	The list items will be numbered with uppercase letters
type="a"	The list items will be numbered with lowercase letters
type="I"	The list items will be numbered with uppercase roman numbers
type="i"	The list items will be numbered with lowercase roman numbers

Numbers:

```
<ol type="1">
    <li>Coffee</li>
    <li>Tea</li>
    <li>Milk</li>
</ol>
```

Uppercase Letters:

```
<oltype="A">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

Lowercase Letters:

```
<oltype="a">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

Uppercase Roman Numbers:

```
<oltype="I">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

```
Lowercase Roman Numbers:
<oltype="i">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

HTML <ol> start Attribute: define the start value of the list.

```
<html>
<body>
   <ol start="50">
      <li>Coffee</li>
      <li>Tea</li>
      <li>Milk</li>
   </ol>
   <ol type="I" start="50">
      <li>Coffee</li>
      <li>Tea</li>
      <li>Milk</li>
   </ol>
</body>
</html>
```

[Task6]. Unordered list in HTML

The unordered list which is created by the **ul Element** that begins with the **<ul>** tag and ends with **</ul>. The attributes required:** ***just*** the **type** attribute.

```
<html>
<body>
   <ul>
      <li>Coffee</li>
      <li>Tea</li>
      <li>Milk</li>
   </ul>
</body>
</html>
```

- **In unordered list, Type Attribute can take one of the next values:**

Disc:

```
<ultype="disc">
   <li>Coffee</li>
   <li>Tea</li>
   <li>Milk</li>
</ul>
```

Circle:

```
<ul type="circle">
   <li>Coffee</li>
   <li>Tea</li>
   <li>Milk</li>
</ul>
```

Square:

```
<ul type="square">
   <li>Coffee</li>
   <li>Tea</li>
   <li>Milk</li>
</ul>
```

None:

```
<ultype="none">
   <li>Coffee</li>
   <li>Tea</li>
   <li>Milk</li>
</ul>
```

[Task7]. The definition list in HTML

The definition list: The list of terms and their meanings is called a **definition list**. This list is created by the **dl Element** that starts with the **<dl>** and ends with **</dl>.** Inside this element two elements are used. The **<dt>** element which goes in front of each term to be defined and a **<dd>** element that is used in front of each definition.

```
<html>
 <body>
<dl>
<dt>Coffee</dt>
<dd>Black hot drink</dd>
<dt>Milk</dt>
<dd>White cold drink</dd>
</dl>
</body>
</html>
```

[Task8]. The nested list in HTML

A Nested List

- Coffee
- Tea
    - Black tea
    - Green tea
- Milk

```
<h2>A Nested List</h2>
<ul>
  <li>Coffee</li>
  <li>Tea
    <ul   type="circle">
  <li>Black tea</li>
      <li>Green tea</li>
    </ul>
  </li>
  <li>Milk</li>
```

Lab (3) : HTML Links and Blocks

❖ HTML Links

HTML links are hyperlinks. You can click on a link and jump to another document. A link does not have to be **text**. It can be an **image** or **any other HTML element.**

In HTML, links are defined with the **<a>** tag:

<a href="*url*">*link text*</a>

href attribute specifies the destination address (http://www.w3schools.com/html/) of the link. The **link text** is the visible part, clicking on the link text will send you to the specified address.

```
<html>
<body>
   <p><a href="http://www.w3schools.com/html/">Visit our HTML tutorial</a></p>
</body>
</html>
```

➢ Absolute and relative Links

1. An absolute URL points to another web site **ex. href="http://www.yahoo.com" A full web address**
2. A relative URL points to a file within a web site **without http**://www.... Such as:
   <a href="html_images.asp">HTML Images</a>
   Or ex. href="/themes/theme.css" if the page in a folder in the same web site.

➢ The target Attribute

The **target** attribute specifies where to open the linked document. The target attribute can have one of the following values:

- **_blank** - Opens the linked document in a new window or tab
- **_self** - Opens the linked document in the same window/tab as it was clicked (this is **default**)

This example will open the linked document in a new browser window/tab:

<a href="http://www.vu.edu.bd/" **target**="_blank">**Visit VU**!</a>

<a href="http://www.vu.edu.bd/" **target**="_self">**Visit VU**!</a>

➢ Going from one Location into another within the Same Web Page

```
<html>
<body>
   <p><a href="#C4">Jump to Chapter 4</a></p>
   <h2>Chapter 1</h2>
   <p>This chapter explains ba bla bla</p>
   <h2>Chapter 2</h2>
   <p>This chapter explains ba bla bla</p>
   <h2>Chapter 3</h2>
   <p>This chapter explains ba bla bla</p>
   <h2 id="C4">Chapter 4</h2>
   <p>This chapter explains ba bla bla</p>
   <h2>Chapter 5</h2>
   <p>This chapter explains ba bla bla</p>
</body>
</html>
```

➢ **Image as Link**

```
<a href=" http://www.bau.edu.jo/">
        <img src="smiley.gif" alt="BAU University" width="42"  height="42"  ">
</a>
```

➢ **Linking to E-mail Addresses**

```
<html>
<body>
   <p>This is an email link:<a href="mailto:asmakhtoom@yahoo.com?Subject=Hello
again">SendMail</a>
   </p>
</body>
</html>
```

❖ HTML Block and Inline Elements

➢ Block-level Elements

A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

The **<div>** element is a block-level element.

```
<html>
<body>
   <div>Hello</div>
   <div>World</div>
   <p>The DIV element is a block element, and will start on a new line.</p>
</body>
</html>
```

- **Inline element**

An inline element does not start on a new line and only takes up as much width as necessary. The **<span>** element is an inline element.

```
<html>
<body>
   <span>Hello</span>
   <span>World</span>
   <p>The SPAN element is an inline element, and will not start on a new line.</p>
</body>
</html>
```

Lab (4) : HTML Tables

❖ Defining an HTML Table

An HTML table is defined with the **<table>** tag. Each table row is defined with the **<tr>** tag. A table header is defined with the **<th>** tag. By default, table headings are bold and centered. A table data/cell is defined with the **<td>** tag.

```
  <table style="width:100%">
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

Note: The **<td>** elements are the data containers of the table.
They can contain all sorts of HTML elements; **text, images, lists, other tables**, etc.

An HTML Table with a Border Attribute

If you do not specify a border for the table, it will be displayed without borders.
A border can be added using the border attribute:

```
<table border="1" style="width:100%">
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

[Task1]. Double Border Table, Coloring and width

```
<html>
<head>
   <style>
      table, th, td {
      border: solid;
      border-width: 3px;  border-color : red;}
      th{color:green;}
      td{color:blue;}
   </style>
</head>
<body>
……… write the code of your table here.
</body>
</html>
```

[Task2]. Single Border Table

```
<head>
   <style>
      table, th, td {
      border: solid ;
      border-collapse: collapse; //single line border
      border-width: 3px;
      border-color : red;}
      th {    color : green;}
      td {    color : blue;}
   </style>
</head>
```

[Task3]. Table Cell Padding and cell spacing

- ✓ Cell padding specifies the space between the cell content and its borders.
- ✓ The cellspacing attribute specifies the space, in pixels, between cells.

```
<table cellpadding="40"  cellspacing=”30”>
……
</table>
```

Or try this

```
<table border="3" bordercolor="red" bgcolor="green"  cellspacing="5"  cellpadding="3">
```

[Task4]. Table caption and cell coloring using bgcolor

- The <caption> tag defines a table caption. The <caption> tag must be inserted **immediately after** the <table> tag. **Note:** You can specify **only one** caption per table.
- bgcolor attribute allow you to give a color to the cell or to the row.
  - ✓ <tr bgcolor="blue">
  - ✓ <th **bgcolor="red"** >

```
<table>
   <caption>Student Marks</caption>
   <tr>
      <th bgcolor="red" >Name</th>
      <th bgcolor="red" >Mark</th>
   </tr>
   <tr>
      <td>Ali</td>
      <td>30</td>
   </tr>
</table>
```

[Task5]. rowspan and colspan Attributes

To combine /merge adjacent cells to create larger cell for data.

- ✓ **rowspan:** This attribute specifies the number of rows a cell should merged.
- ✓ **colspan:** This attribute specifies the number of columns a cell should merged.

Example1: rowspan attribute

```
<table style="width:100%">
  <tr>
    <th>Name:</th>
    <td>Bill Gates</td>
  </tr>
  <tr>
    <th rowspan="2">Telephone:</th>
    <td>555 77 854</td>
  </tr>
  <tr>
    <td>555 77 855</td>
  </tr>
</table>
```

Example2: colspan attribute

```
<table style="width:100%">
  <tr>
    <th>Name</th>
    <th colspan="2">Telephone</th>
  </tr>
  <tr>
    <td>Bill Gates</td>
    <td>555 77 854</td>
    <td>555 77 855</td>
  </tr>
</table>
```

Lab (5) : HTML Forms

❖ The <form> Element

The HTML <**form**> element defines a form that is used to collect user input. An HTML form contains **form elements**. Form elements are different types of input elements, like text fields, checkboxes, radio buttons, submit buttons, and more.

```
<html>
<body>
    <form action="/action_page.php"> First name:<br>
    <input type="text" name="firstname" value="Mickey">
    <br>
    Last name:<br>
    <input type="text" name="lastname" value="Mouse">
    <br><br>
    <input type="submit" value="Submit">
    </form>
    <p>If you click the "Submit" button, the form-data will be sent to a page called
"/action_page.php".</p>
</body>
</html>
```

Form Attributes

Attribute	**Usage**
Action Attribute	The action attribute defines the action to be performed when the form is submitted
Method Attribute	Specifies the HTTP method (GET or POST) to be used when submitting the form data.
Name Attribute	Each input field must have a name attribute to be submitted. If the name attribute is omitted, the data of that input field will not be sent at all.
enctype Attribute	Specifies how the form-data should be encoded when submitting it to the server. can be used only if method="post".

1. Action attribute

Normally, the form data is sent to a web page on the server when the user clicks on the submit button. In the example above, the form data is sent to a page on the server called "/action_page.php". This page contains a server-side script that handles the form data:

<form action="/action_page.php">

If the action attribute is omitted, the action is set to the current page.

2. Method Attribute

The method attribute specifies the **HTTP** method (**GET or POST**) to be used when submitting the form data:

<form action="/action_page.php" method="get">

when **GET** is used, the submitted form data will be visible in the page address field:

/action_page.php?firstname=Mickey&lastname=Mouse

or:

<form action="/action_page.php" method="post">

GET must NOT be used when sending sensitive information! **GET** is best suited for short, non-sensitive, amounts of data, because it has size limitations too.

Always use **POST** if the form data contains sensitive or personal information. The **POST** method does not display the submitted form data in the page address field. **POST** has no size limitations, and can be used to send large amounts of data.

3. Enctype method

Value	**Description**
application/x-www-form-urlencoded	Default. All characters are encoded before sent (spaces are converted to "+" symbols, and special characters are converted to ASCII HEX values)
multipart/form-data	No characters are encoded. This value is required when you are using forms that have a file upload control
text/plain	Spaces are converted to "+" symbols, but no special characters are encoded

❖ The <input> Element

The <**input**> element is the most important form element. The **<input>** element can be displayed in several ways, depending on the **type** attribute.

Type	**Description**
text	Defines a one-line text input field
radio	Defines a radio button (selecting one of many choices)
submit	Defines a submit button (for submitting the form)
checkbox	Defines a checkbox (selecting many choices)
password	Defines a password field (characters are masked)
Button	Defines a clickable button

reset	Defines a reset button (resets all form values to default values)
image	Defines an image as the submit button
file	Defines a file-select field and a "Browse..." button (for file uploads)
hidden	Defines a hidden input field

- **Input Type: text**

<input type="text"> defines a one-line input field for **text input**:

```
<form>
   First name:<br>
   <input type="text" name="firstname"><br>
   Last name:<br>
   <input type="text" name="lastname">
</form>
```

- **Input Type: email**

The **<input type="email">** is used for input fields that should contain an e-mail address.

Depending on browser support, the e-mail address can be automatically validated when submitted.

Some smartphones recognize the email type, and adds ".com" to the keyboard to match email input.

```
<form>
   E-mail:
   <input type="email" name="email">
</form>
```

- **Input Type: password**

<input type="password"> defines a **password field**:

```
<form>
   User name:<br>
   <input type="text" name="username"><br>
   User password:<br>
   <input type="password" name="psw">
</form>
```

- **Input Type: radio**

<input type="radio"> defines a **radio button**.

Radio buttons let a user select ONLY ONE of a limited number of choices:

```
<form>
   <input type="radio" name="gender" value="male" checked> Male<br>
   <input type="radio" name="gender" value="female"> Female<br>
   <input type="radio" name="gender" value="other"> Other
</form>
```

- **Input Type: checkbox**

<input type="checkbox"> defines a **checkbox**.

Checkboxes let a user select ZERO or MORE options of a limited number of choices.

```
<form>
   <input type="checkbox" name="vehicle1" value="Bike"> I have a bike<br>
   <input type="checkbox" name="vehicle2" value="Car"> I have a car
</form>
```

- **Input Type: number**

The **<input type="number">** is used for input fields that should contain a numeric value.

You can set restrictions on the numbers.

Depending on browser support, the restrictions can apply to the input field.

```
<form>
   Quantity (between 1 and 5):
   <input type="number" name="quantity" min="1" max="5"> or
   <input type="number" name="points" min="0" max="100" step="10" value="30">
</form>
```

- **Input Restrictions**

Here is a list of some common input restrictions (some are new in HTML5):

Attribute	Description
disabled	Specifies that an input field should be disabled
max	Specifies the maximum value for an input field
maxlength	Specifies the maximum number of character for an input field
min	Specifies the minimum value for an input field
pattern	Specifies a regular expression to check the input value against
readonly	Specifies that an input field is read only (cannot be changed)
required	Specifies that an input field is required (must be filled out)
size	Specifies the width (in characters) of an input field
step	Specifies the legal number intervals for an input field
value	Specifies the default value for an input field

- **Input Type: date**

The **<input type="date">** is used for input fields that should contain a date.

Depending on browser support, a date picker can show up in the input field.

```
<form>
   Birthday:
   <input type="date" name="bday">
</form>
```

- **Input Type: file**

The **<input type="file">** defines a file select field and a “Browse” button for file upload.

```
<form action="/action_page.php">
   Select a file: <input type="file" name="img">
   <input type="submit">
</form>
```

- **The <textarea> Element**

The **<textarea>** element defines a multi-line input field (**a text area**):

The textarea Element Attributes

- rows Attribute
- cols Attribute

```
<textarea name="message" rows="10" cols="30">
   The cat was playing in the garden.
</textarea>
```

- **The <select> Element (Drop-Down List)**

The **<select>** element defines a **drop-down** list:

```
<select name="cars">
   <option value="volvo">Volvo</option>
   <option value="saab">Saab</option>
   <option value="fiat">Fiat</option>
   <option value="audi">Audi</option>
</select>
```

- **The <button> Element**

The **<button>** element defines a clickable **button**:

```
<button type="button" onclick="alert('Hello World!')">Click Me!</button>
```

- **Input Type: button**

<input type="button"> defines a **button**:

<input type="button" onclick="alert('Hello World!')" value="Click Me!">

- **The Submit Button**

<input type="submit"> defines a button for **submitting** a form to a **form-handler**.

The form-handler is typically a server page with a script for processing input data.

The form-handler is specified in the form's **action** attribute:

```
<form action="action_page.php">
   First name:<br>
   <input type="text" name="firstname" value="Mickey"><br>
   Last name:<br>
   <input type="text" name="lastname" value="Mouse"><br><br>
   <input type="submit" value="Submit">
</form>
```

- **Input Type: reset**

The <input type=”reset”> defines a reset button which resets all form values to its initial values.

```
<form action="/action_page.php">
   Email: <input type="text" name="email"><br>
   Pin: <input type="text" name="pin" maxlength="4"><br>
   <input type="reset" value="Reset">
   <input type="submit" value="Submit">
</form>
```

- **Input Type: hidden**

The <input type=”reset”> defines a hidden input field.

A hidden field let developers include data that cannot be seen or modified by users when a form is submitted.

```
<form action="/action_page.php">
   First name: <input type="text" name="fname"><br>
   <input type="hidden" name="country" value="Norway">
   <input type="submit" value="Submit">
</form>
```

➢ Grouping Form Data with <fieldset>

The **<fieldset>** element groups related data in a form.

The **<legend>** element defines a caption for the <fieldset> element.

```
<form action="action_page.php">
  <fieldset>
    <legend>Personal information:</legend>
    First name:<br>
    <input type="text" name="firstname" value="Mickey"><br>
    Last name:<br>
    <input type="text" name="lastname" value="Mouse"><br><br>
    <input type="submit" value="Submit">
  </fieldset>
</form>
```

Lab (6) : CSS and Bootstrap

- **CSS** stands for **C**ascading **S**tyle **S**heets
- CSS describes **how HTML elements are to be displayed on screen, paper, or in other media**
- CSS **saves a lot of work**. It can control the layout of multiple web pages all at once
- External stylesheets are stored in **CSS files**

The most used css properties used in a web page are as follows:

1) **Color**
2) **Background**
3) **Border**
4) **Margin**
5) **Padding**
6) **Font**
7) **Links**
8) **Lists**
9) **Tables**
10) **Display**
11) **Position**
12) **Float**
13) **Align**
14) **Navigation Bar**
15) **Dropdowns**

➢ Bootstrap

Bootstrap is an open source toolkit for developing with HTML, CSS, and JS. Quickly prototype your ideas or build your entire app with responsive grid system, extensive prebuilt components, and powerful plugins built on jQuery.

Lab (7) : PHP Basics

❖ Basic PHP Syntax

PHP is a **server- side** language which means that PHP script will run on the web server and after execution the **result** will be sent to the browser as xhtml.

PHP Syntax:

<?php …………?>

- ✓ Each line code must end with **;**
- ✓ The file saved with **.php extension.**

❖ Case-Sensitivity in PHP

- ✓ PHP is case sensitive in: **variable names**
- ✓ PHP is not case sensitive in: **function names, keywords and classes names**

❖ Comments in PHP

1. **Single line comment:**
   //…………………………………….

   #................................
2. **Multiple line comment**
   /*………………..

   ………………..*/

❖ Outputting Data to the browser

PHP echo and print Statements

echo and print are more or less the same. They are both used to output data to the screen.

The differences are small: echo has no return value while print has a return value of 1 so it can be used in expressions. echo can take multiple parameters (although such usage is rare) while print can take one argument. echo is marginally faster than print.

1. **print statement:**

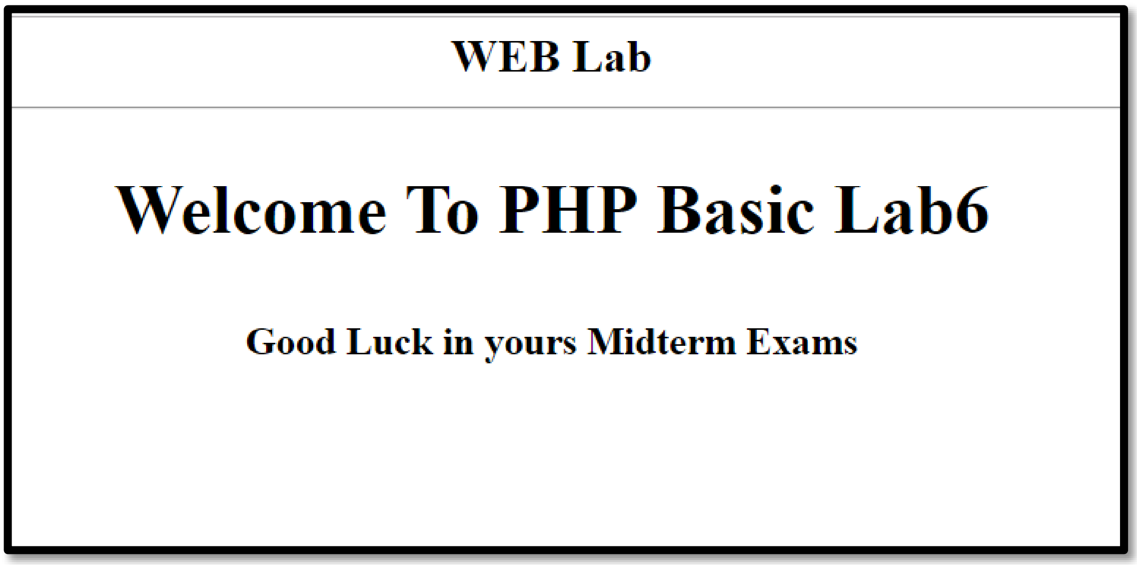
   print (string); is used to print a single string and can contain html tags, the ( ) are **optional**. **Always return 1.**

[Task1]. Write the next code and determine what the output is?

The code	The output

`<?php`
`   print "<hr>";`
`   print "<p><b>V University</b></p>";`
`   print "hello";`
`   print "Welcome";`
` ?>`	

[Task2]. Write the php code to print


WEB Lab

Welcome To PHP Basic Lab6

Good Luck in yours Midterm Exams

2. **echo statement:**

   echo (str1,str2,str3,…..)

   - ✓ Is used to output one or more string and return **nothing**. Can contain html tags.
   - ✓ If it used to print **single** string the ( ) are **optional**.
   - ✓ If it used to print **more** than one string **don't** use (   ).

[Task3]. Write the next code and determine what the output is?

The code	The output

`<?php`
`   echo ("<hr>");`
`   echo ("<p><b>V University</b></p>");`
`   echo "hello<br/>";`
`   echo "One",”Two”,”Three”;`
` ?>`	

❖ Variables in PHP

1. PHP Data Types

You don’t have to determine the data type of a variable when it is declared.

PHP supports the following data types:

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL

2. Declaring Variables.

A variable starts with the **$** sign, followed by the name of the variable. When you assign a **text value** to a variable, put **quotes** around the value.

[Task4]. Write the next code and determine what the output is?

```
<?php
      $txt = "Hello world!";
      $x = 5;
      $y = 10.5;
      $z= true;
      $w=null;
      echo $txt;
      echo "<br>";
      echo $x;
      echo "<br>"
      echo "$x";     // it will not be used as string ,it is ……
      echo "<br>";
      echo $y;
      echo "<br>";
      echo $z;
      echo "<br>";
      echo $w;
?>
```

3. Constants Declaration.

Constants are like variables except that once they are defined they cannot be changed or undefined.

A valid constant name starts with **a letter or underscore** (no $ sign before the constant name) to create a constant, use the define() function.

Syntax

define(*name*, *value*, *case-insensitive*)

- **name:** Specifies the name of the constant
- **value:** Specifies the value of the constant
- **case-insensitive:** Specifies whether the constant name should be case-insensitive. Default is false

Example1. creates a constant with a **case-sensitive** name:

Output

```
<?php
   // case-sensitive constant name
   define("GREETING", "Welcome to VU!");
    echo GREETING;
   echo "GREETING"; // not as variable it will print the string
?>
```

Example2.creates a constant with a **case-insensitive** name:

Output

```
<?php
   // case-insensitive constant name
   define("GREETING","Welcome to VU!", true);
    echo greeting;
?>
```

Constants are Global

Constants can be used across the entire function.

```
<?php
   define("GREETING", "Welcome to W3Schools.com!");
   function myTest() {
      echo GREETING;
   }
   myTest();
?>
```

❖ Concatenation in PHP

You can concat two values together using the dot **(.)**

Example:

```
$txt2 = "WEB Lab";

echo "Study PHP at " . $txt2 . "<br>";
```

❖ Functions in PHP

1. Creating and calling

functions Syntax

```
function functionName() {
    code to be executed;
}
```

- ✓ A user defined function declaration starts with the word "function":
- ✓ A function name can start with a letter or underscore (not a number)
- ✓ Function names are **NOT** case-sensitive

Example1. Function without parameters

Output

```
<?php
  function writeMsg() {
    echo "Hello world!";
  }
  writeMsg(); // call the function
?>
```

Example2. Function with parameters

Output

```
<?php
  function familyName($fname) {
    echo "$fname Nsour.<br>";
  }
  familyName("Ahmad");  familyName("Abdullah");
  familyName("Leen");  familyName("Tamara");
  familyName("Noor");
?>
```

Example3. Function with parameters

```
<?php
   function familyName($fname, $year) {
        echo "$fname Al-Abadi. Born in $year <br>";
   }
   familyName("Ayman", "1975");
   familyName("Jumana", "1978");
   familyName("Fadi", "1983");
?>
```

Output

Example4. Function with parameters

```
<?php

   function countNum($n) {
        echo "$n . <br>";
   }
   for($i=1; $i<=10 ; $i++) countNum($i);
?>
```

Output

Example5. PHP Default Argument Value

The following example shows how to use a default parameter. If we call the function **setHeight()** without arguments it takes the default value as argument:

```
<?php
   function setHeight($minheight = 50) {
        echo "The height is : $minheight <br>";
   }
   setHeight(350);
   setHeight(); //if there is no default it will cause error. setHeight(135);
   setHeight(80);
?>
```

Output

Example6. PHP Functions - Returning values

Output

```
<?php

    function sum($x,$y)
        $z = $x + $y; return $z;
    }
    echo "5 + 10 = " . sum(5, 10) . "<br>";
    echo "7 + 13 = " . sum(7, 13) . "<br>";
    echo "2 + 4 = " . sum(2, 4);
?
```

❖ var_dump() function

The PHP var_dump() function returns **the data type and value**:

Output

```
<?php
    $x =5985;
    var_dump($x);
?>
```

❖ Arrays in PHP

1. Indexed Arrays

Arrays with a numeric index .There are two ways to create indexed arrays:

The index can be assigned automatically (index always starts at 0), like this:

1. $cars = **array**("Volvo", "BMW", "Toyota");

or

2. the index can be assigned manually:

```
$cars[0] = "Volvo";
$cars[1] = "BMW";
$cars[2] = "Toyota";
```

Example1.

```
<?php
   $cars = array("Volvo", "BMW", "Toyota");
   echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2]. ".";
?>
```

Output

➢ **The count Function**

Get The Length of an Array - The count() Function

Example2.

Output

```
<?php
    $cars =array("Volvo","BMW","Toyota");
    echocount($cars);
?>
```

➢ **Loop through an Indexed Array**

To loop through and print all the values of an indexed array, you could use **for** loop.

Example3.

Output

```
<?php
    $cars =array("Volvo","BMW","Toyota");
    $arrlength = count($cars);

    for($x =0; $x < $arrlength; $x++) {  echo$cars[$x];
      echo"<br>";
    }
?>
```

Example4. using for each loop

```
<?php
    $age = array("35","37","43");
    foreach($age as $x) {
        echo        $x;
        echo "<br>";
    }
?>
```

2. Associative Arrays

Associative arrays are arrays that use named keys that you assign to them. There are two ways to create an associative array:

1. $age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

or:

2. $age['Peter'] = "35";

$age['Ben'] = "37";

$age['Joe'] = "43";

The named keys can then be used in a script; key can be **number** or **string**

The value can be of any type.

Example. $color=array(1=>"Red","b"=>"blue","g"=>"green");

Example1.

```
<?php
    $age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
    echo "Peter is " . $age['Peter'] . " years old.";
?>
```

Output

- **Foreach loop through an Associative Array**

To loop through and print all the values of an associative array, you could use a **foreach** loop.

Example2.

```
<?php
    $age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
   foreach($age as $x => $x_value) {
       echo "Key=" . $x . ", Value=" . $x_value; echo "<br>";
   }
?>
```

Output

Lab (8) :Form Handling in PHP

❖ PHP Global Variables - Superglobals

1. **PHP $_SERVER:** $_SERVER is a PHP super global variable which holds information about headers, paths, and script locations.

   $_SERVER['REQUEST_METHOD']: Returns the request method used to access the page (such as POST).
   $_SERVER['PHP_SELF']: Returns the filename of the currently executing script.
2. **PHP $_REQUEST:** is used to collect data after submitting an HTML form.
3. **PHP $_POST:** is widely used to collect form data after submitting an HTML form with method="**post**".
4. **PHP $_GET:** can also be used to collect form data after submitting an HTML form with method="**get**".

❖ Form Handling

Example1.

Part1. Create this web page.

form.html

```
<html>
<head>
   <title>Php contact form</title>
</head>
<body>
   <form name="contact" method="post" action="one.php">
      Name:<input type="text" size=25 name="uname" /><br/>
      Email:<input type="text" size=25 name="email" /><br/>
      Phone:<input type="text" size=25 name="phone" /><br/>
      Message:<textarea name="message" rows="5" cols="35"></textarea><br/>
      <input type="submit" value="Send" name="send"/>
   </form>
</body>
</html>
```

Part2. Create php file called "one.php" in www root to hold the data sent by submit button.
To display the submitted data you could simply echo all the variables.

```
<?php
   $name=$_POST['uname'];
   echo "PHP file received the Name". " ". $name;
 ?>
<br>
Your email address is: <?php echo $_POST["email"]; ?>
```

Example 2:

Modify the **one.php** file as follows:

```
<?php
   if ($_SERVER["REQUEST_METHOD"] == "POST")
      $name= $_POST['uname'];
   else
      $name= $_GET['uname'];
   echo "PHP file received the Name". " ". $name;
 ?>
```

Example 3:

Modify the **one.php** file as follows:

```
<?php
   $name=$_REQUEST['uname'];
   echo $name;
?>
```

❖ Form Validation

Create the next form first in a **"test_form.php" file.**

✓ **Validation rules for the form above are as follows:**

Name	Required Must only contain letters and whitespace
Password	Must consist of more than 8 characters.
Comment	Optional. Multi-line input field (textarea)
Gender	Required. Must select one

Example1:

```
    <html>
    <body>
        <?php
            $nameErr = "";
            $passErr = "" ;
            $genderErr       = "" ;

            $name = "";
            $pass = "";
            $gender= "" ;

            if ($_SERVER["REQUEST_METHOD"] == "POST")
              { if (empty($_POST["name"]))
                    $nameErr = "Name is required";
                else
                    $name= $_POST['name'];

                 if (empty($_POST["pass"]))
                    $passErr = "password is required";
                else
                {
                    $p = $_POST['pass'];
                    if(strlen($p)>8)
                          $pass = $p;
                    else
                       $passErr ="Password must be greater than 8"; }

                 if (empty($_POST["gender"]))
                    $genderErr = "Gender is required";
                 else
                    $gender = $_POST["gender"];}
        ?>

        <h2>PHP Form Validation Example</h2>
        <form method="post" action="<?php echo $_SERVER["PHP_SELF"];?>">
        Name: <input type="text" name="name" value="<?php echo isset($name)?$name:
"";?>"/>
        <font color="red">*<?php echo $nameErr;?></font></p>
        Password: <input type="password" name="pass" value="<?php echo
isset($pass)?$pass: "";?>" />
        <font color="red">*<?php echo $passErr;?></font></p>
```

```
 Gender:
    <input type="radio" name="gender" <?php if (isset($gender)
    && $gender=="female") echo "checked";?> value="female"
    />Female
    <input type="radio" name="gender" <?php if (isset($gender) &&
    $gender=="male") echo "checked";?> value="male" />Male
    <font color="red">* <?php echo $genderErr;?> </font>
    <br>
    <input type="submit" name="submit" value="Send" />
    </form>

    <?php
        echo "<h2>Your Input:</h2>";
        echo $name."<br/>";
        echo $pass."<br/>";
        echo $gender;?>
</body>
</html>
```

❖ File Upload

1. Create The HTML Form

```
<html>
<body>
<form action="upload.php" method="post" enctype="multipart/form-data">
   Select image to upload:
   <input type="file" name="fileToUpload" id="fileToUpload">
   <input type="submit" value="Upload Image" name="submit">
</form>
</body>
</html>
```

2. Create The Upload File PHP Script

The "upload.php" file contains the code for uploading a file:

PHP script explained:

- **$target_dir** = "uploads/" - specifies the directory where the file is going to be placed
- **$target_file** specifies the path of the file to be uploaded
- **$imageFileType** holds the file extension of the file

Note:You will need to create a new directory called "uploads" in the directory where "upload.php" file resides. The uploaded files will be saved there.

```
<?php
$target_dir = "uploads/";
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType = pathinfo($target_file,PATHINFO_EXTENSION);

// Check if image file is a actual image or fake image
if(isset($_POST["submit"])) {
    $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
    if($check !== false) {
        echo "File is an image - " . $check["mime"] . ".";
        $uploadOk = 1;
    } else {
        echo "File is not an image.";
        $uploadOk = 0;
    }
}

// Check if file already exists
if (file_exists($target_file)) {
    echo "Sorry, file already exists.";
    $uploadOk = 0;
}

// Check file size
if ($_FILES["fileToUpload"]["size"] > 500000) {
    echo "Sorry, your file is too large.";
    $uploadOk = 0;
}

// Allow certain file formats
if($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType !=
"jpeg" && $imageFileType != "gif" ) {
    echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";
    $uploadOk = 0;
}
```

```
// Check if $uploadOk is set to 0 by an error
if ($uploadOk == 0) {
    echo "Sorry, your file was not uploaded.";
// if everything is ok, try to upload file
} else {
    if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $target_file)) {
        echo "The file ". basename( $_FILES["fileToUpload"]["name"]). " has been
uploaded.";
    } else {
        echo "Sorry, there was an error uploading your file.";
    }
}
?>
```

Lab (9) : PHP Sessions

Example1

In the example below, we will create a simple page-views counter. The isset() function checks if the "views" variable has already been set. If "views" has been set, we can increment our counter. If "views" doesn't exist, we create a "views" variable, and set it to 1:

view.php

```
<?php
    session_start();
    if(isset($_SESSION['views']))
        $_SESSION['views']= $_SESSION['views']+1;
    else
        $_SESSION['views']= 1;
     echo "Views=". $_SESSION['views'];
?>
<html>
<body>Hello there</body>
</html>
```

The first time you run this script on a **freshly opened browser** the *if statement* will fail because no session variable *views* would have been stored yet. However, if you refreshed the page the *if statement* would be true and the counter would increment by one. Each time you reran this script you would see an increase in *views* by one.

Example2:

First, create the following PHP File:

mypage.php

```
<?php
    // this starts the session
    session_start();
    // this sets variables in the session
    $_SESSION['color']='red';
    // this sets variables in the session
    $_SESSION['color']='red';
    print "Done";
?>
```

Now we are going to make a second page. We again will start with **session_start()** (we need this on every page) - and we will access the session information we set on our first page. Notice we aren't passing any variable; they are all stored in the session.

mypage2.php

```
<?php
   session_start();
   echo "Our color value is ".$_SESSION['color'];
   echo " Our size value is ".$_SESSION['size'];
   echo " Our shape value is ".$_SESSION['shape'];  ?>
```

You can also store an array within the session array. Go back to our mypage.php file and edit it slightly to do this:

```
<?php
    session_start();
   // makes an array
   $colors=array('red', 'yellow', 'blue');
   // adds it to our session
   $_SESSION['color']=$colors;
   $_SESSION['size']='small';
   $_SESSION['shape']='round';
   print "Done"; ?>
```

Now run this on mypage2.php to show our new information:

```
<?php
   session_start();
   //echo a single entry from the array
   echo $_SESSION['color'][2]; ?>
```

```
<?php
   // you have to open the session to be able to modify or remove it
   session_start();
   // to change a variable, just overwrite it
   $_SESSION['size']='large';
   //you can remove a single variable in the session
   unset($_SESSION['shape']);
   // or this would remove all the variables in the session, but not the session //itself
   session_unset();
   // this would destroy the session variables
   session_destroy(); ?>
```

Note: By default, a session lasts until the user closes the browse

Lab (10) :Databases with SQL and PHP

❖ Database Creation

- ➢ Create database called Persons.
- ➢ Create Table called contacts like this:

ID (PK) integer	FirstName (varchar)	LastName (varchar)	PhoneNumber (varchar)	Email (varchar)

- ➢ Insert three records in your table
- ➢ Use the database in the next php code.

❖ Create the main page of the user interface (index.html)

```
<html>
<head><title>Persons: Main Page </title></head>
<body>
   <h1>Persons Information Application </h1>
   <p> Welcome to my first Web Application </p>
   <p> Next is the main menu of this Application
   <ul>
      <li><a href="./index.html">Home page </a></li>
      <li><a href="./show_all.php">Show all contacts </a></li>
   </ul>
   </p>
</body>
</html>
```

❖ Creating the first page: Showing all contacts (show_all.php)

```
<html>
<head><title>Persons: Show All Contacts</title></head>
<body>
   <h1>Persons Application: All my contacts </h1>
   <p>Next is a alist of all my contacts </p>
   <?php
   $con=mysqli_connect("localhost","root","","Persons");
   // Check connection
   if (mysqli_connect_errno()) {
      echo "Failed to connect to MySQL: " . mysqli_connect_error(); }
```

```
    else{
        echo "Connected successfully"; }
    $result = mysqli_query($con,"SELECT * FROM contacts");
    while($row = mysqli_fetch_array($result)){
        echo $row['FirstName'] . " " . $row['LastName'];
        echo "<br>";
    }
    mysqli_close($con);
    ?>
    <br /><hr />
    <p> Go back to <a href="./index.html">Home page </a></p>
</body>
</html>
```

❖ **Display the Result in an HTML Table**

```
<html>
<head><title>Persons: Show All Contacts</title></head>
<body><h1>Persons Application: All my contacts </h1>
    <p>Next is a alist of all my contacts </p>
    <?php
    $con=mysqli_connect("localhost","root","","Persons");
    // Check connection
    if (mysqli_connect_errno())  {
        echo "Failed to connect to MySQL: " . mysqli_connect_error(); }
    else{
        echo "Connected successfully"; }
    $result = mysqli_query($con,"SELECT * FROM contacts");
    echo "<table border='1'>
    <tr>
        <th>Firstname</th>
       <th>Lastname</th>
       <th>PhoneNumber</th>
    </tr>";
    while($row = mysqli_fetch_array($result)){
        echo "<tr>";
            echo "<td>" . $row['FirstName'] . "</td>";
            echo "<td>" . $row['LastName'] . "</td>";
            echo "<td>" . $row['PhoneNumber'] . "</td>";
       echo "</tr>";
    }
    echo "</table>";
    mysqli_close($con); ?>
    <br /><hr />
    <p> Go back to <a href="./index.html">Home page </a></li></p>
</body>
</html>
```

❖ Insert Data from a Form into a Database

Example:

We will create an HTML form that can be used to add new records to the "Persons" table.

```
<html>
<body>
    <form name="new_form" method="post" action="insert.php" >
        Firstname: <input type="text"    name="firstname" />  Lastname: <input type="text"
name="lastname" />
        Age: <input type="text" name="age" />
        <input type="submit" />
    </form>
</body>
</html>
```

```
<?php
    $con=mysqli_connect("localhost","root","","Persons");
    // Check connection
    if (mysqli_connect_errno())
        echo "Failed to connect to MySQL: " . mysqli_connect_error();
    $firstname = $_POST['firstname'];
    $lastname =$_POST['lastname'];
    $age =         $_POST['age'];
    $sql="INSERT INTO Persons (FirstName, LastName, Age)  VALUES ('$firstname',
'$lastname', $age)";
    if (!mysqli_query($con, $sql))
        die('Error: ' . mysqli_error($con));
    echo "1 record added";
    mysqli_close($con);
?>
```

Lab (11) : PHP and MySql CRUD

1. Create database using PHP code.

```
<?php
   $servername = "localhost";
   $username = "username";
   $password = "password";

   // Create connection
   $conn = new mysqli($servername, $username, $password);
   // Check connection
   if ($conn->connect_error) {
       die("Connection failed: " . $conn->connect_error);
   }

   // Create database
   $sql = "CREATE DATABASE myDB";
   if ($conn->query($sql) === TRUE) {
       echo "Database created successfully";
   } else {
       echo "Error creating database: " . $conn->error; }

$conn->close();
?>
```

2. Create a MySQL Table Using MySQLi PHP

```
<?php
   $servername = "localhost";
   $username = "username";
   $password = "password";
   $dbname = "myDB";

   // Create connection
   $conn = new mysqli($servername, $username, $password, $dbname);
   // Check connection
   if ($conn->connect_error) {
       die("Connection failed: " . $conn->connect_error);
   }
```

```
    // sql to create table
    $sql = "CREATE TABLE Students (
        id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
        firstname VARCHAR(30) NOT NULL,  lastname VARCHAR(30) NOT NULL,
         email  VARCHAR(50), reg_date TIMESTAMP   )";

    if ($conn->query($sql) === TRUE) {
        echo "Table Students created successfully";
    } else {
        echo "Error creating table: " . $conn->error;
    }
    $conn->close();
?>
```

3. Insert Data into MySQL

```
<?php
    $servername = "localhost";
    $username = "username";
    $password = "password";
    $dbname = "myDB";

    // Create connection
    $conn = new mysqli($servername, $username, $password, $dbname);
    // Check connection
    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }

    $sql = "INSERT INTO Students (firstname, lastname, email)  VALUES ('John', 'Doe',
'john@example.com')";

    if ($conn->query($sql) === TRUE) {
        echo "New record created successfully";
    } else {
        echo "Error: " . $sql . "<br>" . $conn->error;
    }

$conn->close();
?>
```

4. Insert Multiple Records into MySQL

```
<?php
    $servername = "localhost";
    $username = "username";
    $password = "password";
    $dbname = "myDB";

    // Create connection
    $conn = new mysqli($servername, $username, $password, $dbname);
    // Check connection
    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }

    $sql = "INSERT INTO Students (firstname, lastname, email)  VALUES ('John', 'Doe',
'john@example.com');";
    $sql .= "INSERT INTO Students (firstname, lastname, email)  VALUES ('Mary', 'Moe',
'mary@example.com');";
    $sql .= "INSERT INTO Students (firstname, lastname, email)  VALUES ('Julie', 'Dooley',
'julie@example.com')";

    if ($conn->multi_query($sql) === TRUE) {
         echo "New records created successfully";
    } else {
        echo "Error: " . $sql . "<br>" . $conn->error; }
    $conn->close();
?>
```

5. Select Data from a MySQL Database

```
<?php
    $servername = "localhost";
    $username = "username";
    $password = "password";
    $dbname = "myDB";

    // Create connection
    $conn = new mysqli($servername, $username, $password, $dbname);
    // Check connection
    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error); }

    $sql = "SELECT id, firstname, lastname FROM Students ";
    $result = $conn->query($sql);
    if ($result->num_rows > 0) {
        echo "<table border= '1'>
    <tr>
        <th>ID</th>
        <th>FirstName</th>
        <th>LastName</th>
        <th>Action</th>
    </tr>";

    // output data of each row
    while($row = $result->fetch_assoc()) {
    echo "<tr>
            <td>". $row['id'] . "</td>
            <td>". $row['firstname'] . "</td>
            <td>". $row['lastname'] . "</td>
            <td><a href='update.php?id=".$row['id']."'>Update</a>/<a
href='delete.php?id=".$row['id']."'>Delete</a></td>
            </tr>";
    }
    echo "</table>";
    } else {
        echo "0 results";
    }
    $conn->close();
?>
```

6. Delete Data from a MySQL Table

```
<?php
    $servername = "localhost";
    $username = "username";
    $password = "password";
    $dbname = "myDB";

    // Create connection
    $conn = new mysqli($servername, $username, $password, $dbname);
    // Check connection
    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }

    $id=$_GET["id"];
    // sql to delete a record
    $sql = "DELETE FROM Students WHERE id='$id' ";
    if ($conn->query($sql) === TRUE) {
        echo "Record deleted successfully";
    } else {
    echo "Error deleting record: " . $conn->error;
    }
    $conn->close();
?>
```

7. Update Data in a MySQL Table

```
<?php
    $servername = "localhost";
    $username = "username";
    $password = "password";
    $dbname = "myDB";

    // Create connection
    $conn = new mysqli($servername, $username, $password, $dbname);
    // Check connection
    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }
    $id=$_GET["id"];

    $sql = "UPDATE Students SET lastname='Doe' WHERE id= id='$id' ";
    if ($conn->query($sql) === TRUE) {
        echo "Record updated successfully";
    } else {
        echo "Error updating record: " . $conn->error;
    }
    $conn->close();
?>
```

Lab (12) : FINAL LAB EXAMINATION

Task 1: Quiz

Task 2: Two Lab Problem Solving. (Include GUI)

Task 3: Viva Voce

Task 4: Project Demonstration