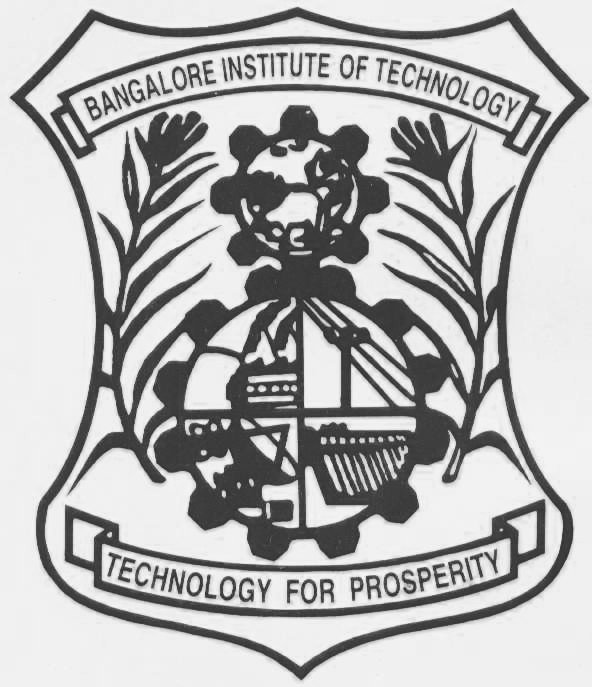
BANGALORE INSTITUTE OF TECHNOLOGY

****

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

**WEB PROGRAMMING LABORATORY**

**(06CSL78)**

**LABORATORY CERTIFICATE**

****

This is to certify that the **WEB PROGRAMMING LABORATORY** has been satisfactorily completed by **Vishal P Uchil,** bearing the **USN 1BI09IS057,** as prescribed by the Visvesvaraya Technological University during the VII Semester in the academic year 2012-13.

Marks Awarded: \_\_\_\_\_\_\_/25 Date:

Dr. T Asha Miss. M Chetana Dr. J Prakash

Staff In-Charge Staff In-Charge Head of Dept.,

Associate Prof. Asst. Prof. I.S.E., B.I.T.

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| 1. | Develop and demonstrate a XHTML document that illustrates the use external style sheet, ordered list, table, borders, padding, colour, and the <span> tag. |
| 2. | Develop and demonstrate a XHTML file that includes Javascript script for the following problems:  a) Input : A number n obtained using prompt  Output : The first n Fibonacci numbers  b) Input : A number n obtained using prompt  Output : A table of numbers from 1 to n and their squares using alert |
| 3. | Develop and demonstrate a XHTML file that includes Javascript script that uses functions for the following problems:  a) Parameter : A string  Output : The position in the string of the left-most vowel  b) Parameter : A number  Output : The number with its digits in the reverse order |
| 4. | a) Develop and demonstrate, using Javascript script, a XHTML document that collects the USN ( the valid format is: A digit from 1 to 4 followed by two upper-case characters followed by two digits followed by two upper-case characters followed by three digits; no embedded spaces allowed) of the user. Event handler must be included for the form element that collects this information to validate the input. Messages in the alert windows must be produced when errors are detected.  b) Modify the above program to get the current semester also (restricted to be a number from 1 to 8) |
| 5. | a) Develop and demonstrate, using Javascript script, a XHTML document that contains three short paragraphs of text, stacked on top of each other, with only enough of each showing so that the mouse cursor can be placed over some part of them. When the cursor is placed over the exposed part of any paragraph, it should rise to the top to become completely visible.  b) Modify the above document so that when a paragraph is moved from the top stacking position, it returns to its original position rather than to the bottom. |

|  |  |
| --- | --- |
| 6. | a) Design an XML document to store information about a student in an engineering college affiliated to VTU. The information must include USN, Name, Name of the College, Brach, Year of Joining, and e-mail id. Make up sample data for 3 students. Create a CSS style sheet and use it to display the document.  b) Create an XSLT style sheet for one student element of the above document and use it to create a display of that element. |
| 7. | a) Write a Perl program to display various Server Information like Server Name, Server Software, Server protocol, CGI Revision etc.  b) Write a Perl program to accept UNIX command from a HTML form and to display the output of the command executed. |
| 8. | a) Write a Perl program to accept the User Name and display a greeting message randomly chosen from a list of 4 greeting messages.  b) Write a Perl program to keep track of the number of visitors visiting the web page and to display this count of visitors, with proper headings. |
| 9. | Write a Perl program to display a digital clock which displays the current time of the server. |
| 10. | Write a Perl 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. |
| 11. | Write a PHP program to store current date-time in a COOKIE and display the ‘Last visited on’ date-time on the web page upon reopening of the same page. |
| 12. | Write a PHP program to store page views count in SESSION, to increment the count on each refresh, and to show the count on web page. |
| 13. | Create a XHTML form with Name, Address Line 1, Address Line 2, and E-mail text fields. On submitting, store the values in MySQL table. Retrieve and display the data based on Name. |
| 14. | Using PHP and MySQL, develop a program to accept book information viz. Accession number, title, authors, edition and publisher from a web page and store the information in a database and to search for a book with the title specified by the user and to display the search results with proper headings. |

**Instructions to run programs:**

PROGRAMS 1-5 are HTML/XHTML programs.

* Save the file in the respective format (Ex: .html, .css, etc).
* Save all files from the same program under the same directory.
* To run the program, double click on the .html file.

PROGRAMS 6a & 6b are XML/XSL programs.

* Save both the files under same folder.
* To run program, execute the XML file (.xml extension).

PROGRAMS 7-10 have Pearl scripts.

* Save the Pearl file (.pl extension) in “C:\wamp\bin\apache\Apache2.2.11\cgi-bin\vishal\”.
* Start the WAMP.
* Go to “localhost” in browser and execute the Pearl program by typing the file name along with its path from “cgi-bin” onwards in the URL. Ex:- “localhost/cgi-bin/vishal/7a.pl”.
* Suppose the program has an HTML file as part of it, execute the HTML file to run the program.
* Single line comments in Perl start with ‘#’.

PROGRAMS 11-14 have PHP scripts.

* Save the PHP file (.php extension) in “C:\wamp\www\vishal\”.
* Start the WAMP.
* Go to “localhost” in browser and execute the PHP program by typing the file name along with its path from its parent directory in the URL. Ex:- “localhost/ vishal/11.php”.
* Suppose the program has an HTML file as part of it, execute the HTML file to run the program.
* Single line comments in PHP start with ‘#’ or ‘//’.

For all MySQL database programs (PROGRAMS 10, 13 & 14)

* Click on the WAMP icon in the toolbar (bottom-right of desktop).
* Select ‘MySQL Console’ under ‘MySQL’.
* Enter the password as prompted (Default password : ise123).
* Create your database (‘create database vishal;’).
* Use your database (‘use vishal;’).
* Create table as required.
* Entries made in HTML will be reflected under MySQL.

**HTML**

HTML stands for **HyperText Markup Language**. Developed by scientist Tim Berners-Lee in 1990, HTML is the "hidden" code that helps us communicate with others on the World Wide Web (WWW).

It is a standardized system for tagging text files to achieve font, colour, graphic, and hyperlink effects on World Wide Web pages

When writing HTML, you add "tags" to the text in order to create the structure. These tags tell the browser how to display the text or graphics in the document.

**XHTML**

XHTML (Extensible Hypertext Markup Language) is a family of XML markup languages that mirror or extend versions of the widely used Hypertext Markup Language (HTML), the language in which web pages are written.

XHTML is extremely similar to HTML but follows the rules of XML. The main differences between HTML and XHTML are the case-sensitivity, the need to use closing tags for all tags, the need to use quotes around all attributed values and that all attribute must be in lower case.

HTML is an application of SGML (Standard Generalized Markup Language), and allows an author to omit certain tags and use attribute minimization.2 The Extensible HyperText Markup Language, or XHTML, is an application of XML (Extensible Markup Language). It doesn’t permit the omission of any tags or the use of attribute minimization. However, it provides a shorthand notation for empty elements—for example, we could use <br/> instead of <br></br>—which HTML does not. A conforming XML document must be well formed, which, among other things, means that there must be an end tag for every start tag, and that nested tags must be closed in the right order.

**DHTML**

DHTML is a combination of technologies to make web pages dynamic. To most people, DHTML means a combination of HTML 4.0, Style Sheets and JavaScript. It is commonly used to describe advanced JavaScript.

**XML and HTML**

XML was designed to transport and store data. XML focuses on what data is.

HTML was designed display data and focuses on how data looks.

**XML**

Extensible Markup Language (XML) is a markup language created to structure, store, and transport data by defining a set of rules for encoding documents in a format that is both human-readable and machine-readable.

The design goals of XML emphasize simplicity, generality and usability over the Internet. It is a textual data format with a strong support via Unicode for the languages of the world. Although the design of docs focuses on docs, it is widely used for the representation of arbitrary data structures.

**XSLT**

XSLT (Extensible Stylesheet Language Transformations) is a language for transforming XML documents into other XML documents,[1] or other objects such as HTML for web pages, plain text or into XSL Formatting Objects which can then be converted to PDF, PostScript and PNG. The original document is not changed; rather, a new document is created based on the content of an existing one.

XSLT processing model involves

* One or more XML source docs
* One or more XSLT stylesheet modules
* XSLT template processing engine
* Resultant docs

Apps often use XSLT to convert XML data into HTML or XHTML docs for display of web pages.

**PERL**

Perl is a high-level, general-purpose, interpreted, dynamic programming language.

Though Perl is not officially an acronym, there are various backronyms in use, such as: Practical Extraction and Reporting Language.

The language provides powerful text processing facilities without the arbitrary data length limits of many contemporary Unix tools, facilitating easy manipulation of text files. Perl gained widespread popularity in the late 1990s as a CGI scripting language, in part due to its parsing abilities.

Perl is often used as a glue language, tying together systems and interfaces that were not specifically designed to interoperate, and for "data munging",[53] that is, converting or processing large amounts of data for tasks such as creating reports. In fact, these strengths are intimately linked. The combination makes Perl a popular all-purpose language for system administrators, particularly because short programs can be entered and run on a single command line.

Perl code can be made portable across Windows and Unix

**PHP**

PHP is an open source general-purpose server-side scripting language originally designed for Web development to produce dynamic Web pages. It is one of the first developed server-side scripting languages to be embedded into an HTML source document rather than calling an external file to process data. The code is interpreted by a Web server with a PHP processor module which generates the resulting Web page. It also has evolved to include a command-line interface capability and can be used in standalone graphical applications.[2] PHP can be deployed on most Web servers and also as a standalone shell on almost every operating system and platform free of charge.

While PHP originally stood for Personal Home Page, it is now said to stand for PHP: Hypertext Preprocessor, a recursive acronym.

**MySQL**

MySQL is the world's most used open source relational database management system (RDBMS) as of 2008 that runs as a server providing multi-user access to a number of databases. The SQL phrase stands for Structured Query Language.

MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack (and other 'AMP' stacks).

It is compatible with all Operating Systems. It is light-weight, provides multi-language support and is 100% remotely configurable.

**Common HTML Tags**

|  |  |  |
| --- | --- | --- |
| **Tag** | **What it is** | **When to use it** |
| **<a>** | Anchor (most commonly a link) | Vital. Use to create links in content. Use the title attribute whenever the contents of the **<a>…</a>** pair do not accurately describe what you’ll get from selecting the link. Title attribute often displays as a tooltip in visual browsers, which may be a helpful usability aid. |
| **<abbr>** | Defines an abbreviation | Works in a similar way to **<dfn>** and**<acronym>**, using a **title** attribute (displays a tooltip in standard visual browsers). e.g. **<abbr title=”Hypertext markup language”>HTML</abbr>** |
| **<ACRONYM>** | Defines an acronym | Works in a similar way to **<abbr>**and **<dfn>**, using a **title** attribute (displays a tooltip in standard visual browsers). |
| **<ADDRESS>** | Used for marking up a physical (e.g. mailing) address | Not commonly used. Recommend looking into microformats, which allow for more detail and interoperability. |
| **<APPLET>** | Inserts a Java applet | The old way to insert a Java app. Use**<object>** instead today. |
| **<AREA>** | Hotspot in image map | Avoid image maps where possible. Occasionally necessary. |
| **<BASE>** | Specifies the base location of the document. | Use only when necessary. Adjusts any relative links and paths within the document. |
| **<BASEFONT>** | Sets default font size | Display info – never use it |
| **<BIG>** | Larger text | Display info – never use it |
| **<BLINK>** | Makes text blink | You go to hell if you use this |
| **<BLOCKQUOTE>** | Large quoted block of text | Use for any quoted text that constitutes one or more paragraphs (note: should contain <p> tags as well). Use **<q>** for quotations within a paragraph. Often used in conjunction with **<cite>** to cite the quotation’s source. |
| **<BODY>** | Document body | Essential (unless you’re using frames) |
| **<BR>** | Line break | This is arguably display information. Still in common use, but use with restraint. |
| **<B>** | Bold text | Display info – never use it |
| **<BUTTON>** | Used for a standard clickable button within a form | Often better than **<input type=”button” />** or **<input type=”submit” />**, as it allows you to assign different styles based on the HTML element alone, whereas differentiating style based on the type of input is less well supported. |
| **<CAPTION>** | Caption for a table: describes the table’s contents | The correct way to assign a title to a table |
| **<CENTER>** | Centred block | Display info – never use it. Use**<div>** or some other block-level tag with the style text-align:center instead |
| **<CITE>** | Defines a citation | Defines the source of a quotation (in conjunction with content in **<q>** or**<blockquote>** pairs). |
| **<CODE>** | Defines an extract of code | Not commonly used. Similar to**<pre>** tag, but collapses consecutive white spaces and line breaks in the source. |
| **<COL>** | Identifies a particular column in a table | Can be very useful. e.g. **<col class=”namecol”>** can be applied to each first column in a series of tables, then the width of each column may be set to be equal in the stylesheet, overriding the table’s natural tendency to adjust its own column widths to fit its contents. |
| **<DFN>** | Definition of a term | Works in a similar way to **<abbr>**and **<acronym>**, using a **title**attribute (displays a tooltip in standard visual browsers). |
| **<DIR>** | Directory list | Now deprecated. Use a standard**<ul>** or other list instead. |
| **<DIV>** | Division | Specifies a logical division within a document. Use it to separate or identify chunks of content that are not otherwise distinguished naturally using other tags.  One of the most common HTML tags. |
| **<DL>** | Definition list | Contains one or more definition-term / definition-description pairs. |
| **<DT>** | Definition term | Used as part of a **<dt></dt><dd></dd>** pair within a definition list (**<dl></dl>**) |
| **<DD>** | Definition description |
| **<EM>** | Emphasis | Commonly used in place of the old**<i>**(italics) tag to indicate emphasis (but less than **<strong>**) |
| **<FONT>** | Font settings | Display info – never use it |
| **<FORM>** | Input form | Essential for data input |
| **<H1>** | Level 1 header | Aim to have one H1 on each page, containing a description of what the page is about. |
| **<H2>** | Level 2 header | Defines a section of the page |
| **<H3>** | Level 3 header | Defines a sub-section of the page (should always follow an H2 in the logical hierarchy) |
| **<H4>** | Level 4 header | Etc. Less commonly used |
| **<H5>** | Level 5 header | Less commonly used. Only complex academic documents will break down to this level of detail. |
| **<H6>** | Level 6 header | Less commonly used |
| **<HEAD>** | Document head | Essential. Contains information about a page that does not constitute content to be communicated as part of the page. |
| **<HR>** | Horizontal rule | Display info with no semantic value – never use it. “Horizontal”, by definition, is a visual attribute. |
| **<HTML>** |  | Core element of every web page. |
| **<IMG >** | Show an image | Vital. Always use the **alt** or **longdesc**attributes when the image has content value |
| **<INPUT>** | Input fields within forms | Vital. (I prefer to use **<button>** for buttons and submit buttons though) |
| **<ISINDEX>** | Old type of search input | Not really used any more. Use**<form>** instead. |
| **<I>** | Italicised text | Display info – never use it |
| **<KBD>** | Keyboard input | Display info – never use it |
| **<LINK>** | Defines a relationship to another document | Commonly used to reference external stylesheets, but has other minor uses |
| **<LI>** | List item | Specifies an item in an unordered or ordered list (**<ul>** or **<ol>**) |
| **<MAP>** | Client-side imagemap | May have occasional value, but only use when absolutely necessary |
| **<MARQUEE>** | Makes text scroll across the screen | See **<blink>** |
| **<MENU>** | Menu item list | Deprecated. Do not use. Use other standard list types instead. |
| **<META>** | Meta-information | Useful way to insert relevant information into the <head> section of the page that does not need to be displayed. |
| **<OL>** | Ordered list | Type of list where the order of elements has some meaning. Generally rendered with item numbers (best managed with CSS). |
| **<OPTION>** | Selection list option | Vital for options within a drop-down control. |
| **<PARAM>** | Parameter for Java applet | Used in conjunction with an**<object>** or **<applet>** tag to pass additional setting information at runtime. |
| **<PRE>** | Preformatted text | Renders text in a pre-formatted style, preserving line breaks and all spaces present in the source. May be useful. *(This one’s a paradox, as it is strictly display info that applies only to visual browsing, but it’s still so commonly used and useful that I’m hesitant to advise against using it.)* |
| **<P>** | Paragraph | Only use to denote a paragraph of text. Never use for spacing alone. |
| **<Q>** | Short quotation | Use for inline quotations (whereas**<blockquote>** should be used for quotations of a paragraph or more). Often used in conjunction with**<cite>** to cite the quotation’s source. |
| **<SAMP>** | Denotes sample output text | Similar to the <code> tag. Rarely used. Avoid. |
| **<SCRIPT>** | Inline script (e.g. JavaScript) | It’s better to have all scripts as separate files than to write inline or in the **<head>**section, however still has its uses. |
| **<SELECT>** | Selection list | A drop-down selector for a form. |
| **<SMALL>** | Smaller text | Display info – never use it |
| **<SPAN>** | An inline span within text | Use to apply meaning (and style) to a span of text that goes with the flow of content (whereas a **<div>** tag is block-level and breaks the flow) |
| **<Strikeout>** |  | Display info – never use it |
| **<STRONG>** | Strong emphasis | Use this instead of the old **<b>** tag. |
| **<STYLE>** | CSS style settings | Normally used in **<head>** section of a page. Try to use external stylesheets, to enable you to apply different styles for different output media. |
| **<SUB>** | Subscript text | Arguably display info – recommend using alternative tags (e.g. **<cite>**). May be required in some academic uses, e.g. Chemical formulas. |
| **<SUP>** | Superscript text |
| **<TABLE>** | Table | Use for repeated data that has a naturally tabular form. Never use for layout purposes. |
| **<TD>** | Table data cell | A cell containing actual data. If a cell actually contains a descriptor or identifier for a row or column, use a**<th>** (table header) tag, not a**<td>.** This usually applies to column headers (within a **<thead>**), column footers (within a **<tfoot>**), as well as row headers (usually the first cell in a row in the **<tbody>**). |
| **<TEXTAREA>** | Multi-line text input area in a form | Essential |
| **<TH>** | Table column or row header cell | May appear in a **<thead>** (to denote a column header cell), **<tbody>** (to denote a row header), and in**<tfoot>** (to denote a column foot cell, e.g. a total) |
| **<TBODY>** | Indicates the main body of a data table | It is always worth using this tag, as well as using **<thead>** and **<tfoot>**where appropriate.  Note that it is permissible to have more than one **<tbody>**, **<thead>**, and **<tfoot>** in the same table. |
| **<THEAD>** | The head section of a table | The place to put column header cells (**<th>**) |
| **<TFOOT>** | The foot section of a table | Good place to put e.g. summary data, such as totals. Note that it goes before the **<tbody>** tag! |
| **<TITLE>** | Document title | Essential |
| **<TR>** | Table row | Essential with tables |
| **<TT>** | “Teletype” - simulates typewriter output | Similar to <pre>, except that it collapses white space like normal HTML (whereas <pre> leaves all consecutive white space intact). Avoid if possible |
| **<UL>** | Unordered list | Essential. Use for lists where the order or items has no particular importance. |
| **<U>** | Underline text | Display info – never use it |
| **<VAR>** | Variable in computer code | Obscure tag, may only be useful in academic documents. Avoid. |

**PROGRAM 1**

Develop and demonstrate a XHTML document that illustrates the use external style sheet, ordered list, table, borders, padding, colour, and the <span> tag.

**//p1.html**

<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE html PUBLIC "\_//W3C//DTD XHTML 1.1//EN"

"http://www.w3.org/TR/xhtml11/DTD/xhtml.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<link rel="stylesheet" type="text/css" href="mystyle.css"/> <title> LAB PROGRAM! </title>

</head>

<body>

<h1> Information Science & Engg Dept </h1> <h1>Bangalore Institute of Technology </h1> <h1>K.R.road</h1>

<h2>This header is blue</h2>

<p>This paragraph has a left margin of 50 pixels</p> <table border="4" width="5%">

<tr>

<th width="500"> NAMES </th> <th> Email</th>

</tr>

<tr>

<td width="500"> MR RN </td>

<td> nagarajar\_ise\_dept@yahoo.com</td>

</tr>

<tr>

<td width="500"> Asha</td>

<td> ash@yahoo.com</td>

</tr>

<tr>

<td width="500"> Mr JP</td>

<td>jprakash@yahoo.com</td>

</tr>

<tr>

<td width="500">Mr.Shivkumar</td>

<td>shivsp@yahoo.com</td>

</tr>

</table>

<hr> <!--HORIZONTAL LINE--> <ol> <!--Ordered list-->

<h1><li>M.Shilpa</li></h1>

<h1><li>M.Chetana</li></h1>

<h1><li>v.vani</li></h1>

<h1><li>padmanabh</li></h1>

</ol>

<hr>

<p>

<br>

<span>Information Science</span>

</br>

<span>BIT</span>

<br><span>KR ROAD</span></br>

</p>

</body>

</html>

**//Mystyle.css**

p,table,li

{

font-family:"lucida calligraphy",comic sans,'sans serif'; margin-left: 5pt;

}

p { word-spacing: 5px;}

body { background-color:rgb(200,255,205);} p,li,td( font-size: 75%;}

td { padding: 0.5cm ;}

th {

text-align:center;

font-size:85%;

}

h1,h2,h3,hr {text-align:center;color:#483d8b;}

table

{

border-style:outset;

background: rgb(100,255,255);

}

li {list-style-type:lower-roman;}

span

{

text-align:center;

color:white;

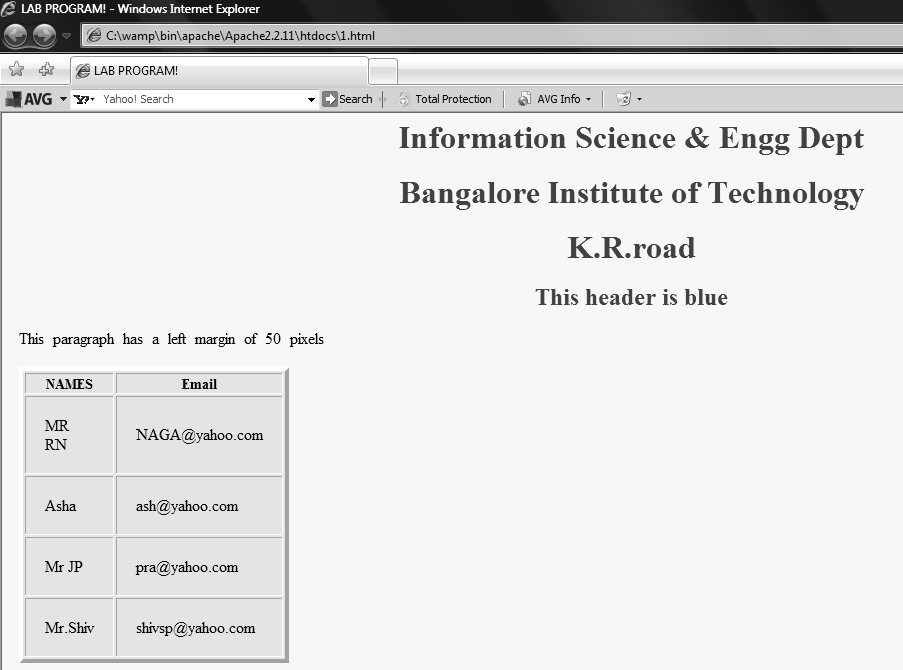
background-color:blue; font-size:29pt;

font style:italic;

font-weight:bold;

}

**OUTPUT:**



**PROGRAM 2**

Develop and demonstrate a XHTML file that includes Javascript script for the following problems:

a) Input : A number n obtained using prompt

Output : The first n Fibonacci numbers

b) Input : A number n obtained using prompt

Output : A table of numbers from 1 to n and their squares using alert

**//p2a.html**

<?xml version ="1.0"encoding="utf-8"=/>

<!DOCTYPE html PUBLIC="-/W3C//DTD XHTML1.1//EN"

"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd>

<html xmlns="http://ww.w3.org/1999/xhtml">

<head>

<script>

function f(n)

{ var s = 0;

if(n== 0) return(s);

if(n ==1)

{

s += 1;

return(s);

}

else

{

return(f(n-1) + f(n-2));

}

}

function show(n)

{ var i;

for(i = 0;i < n;i++)

{

f1.s.value += " "+f(i);

}

}

</script>

</head>

<body bgcolor="#123456" text="white">

<form name=f1>

Computations:<input type=text size=4 name="n">

<input type=button value="Compute" onclick="show(f1.n.value);">

<input type=reset value="Reset"><br>

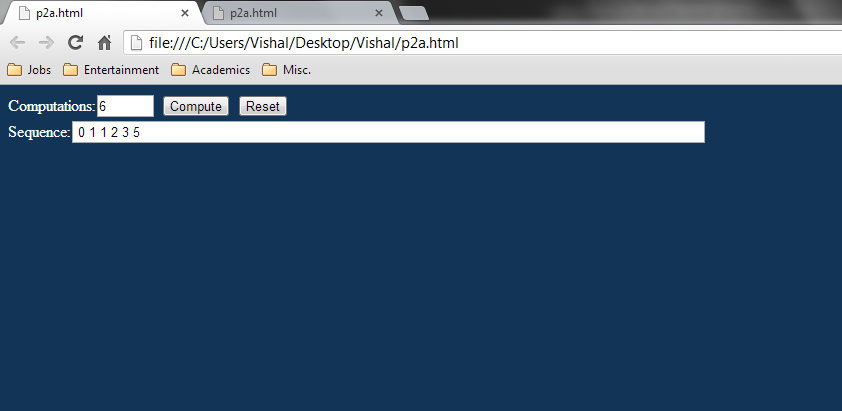
Sequence:<input type=text size="100" name="s">

</form>

</body>

</html>

**OUTPUT:**



**//p2b.html**

<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"

"http://www.w3.org/TR/xhtml 11/DTD/xhtml 11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title> Program 2b </title>

<body>

<script type = "text/javascript">

var num = prompt("Enter a number:\n"," ");

if(num > 0 && num != null)

{

msgstr = "number and its squares are\n";

for(i = 1;i <= num;i++)

msgstr = msgstr + i +"-" + i \* i + "\n";

alert(msgstr);

}

else

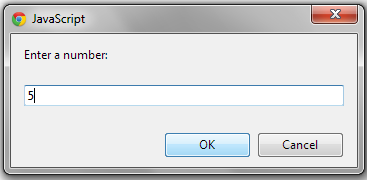
alert("no input supplied");

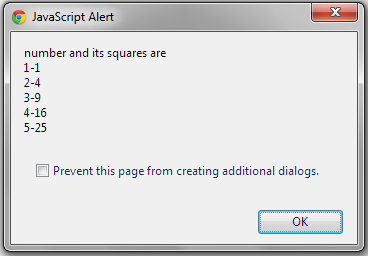
</script>

</body>

</html>

**OUTPUT:**





**PROGRAM 3**

Develop and demonstrate a XHTML file that includes Javascript script that uses functions for the following problems:

a) Parameter : A string

Output : The position in the string of the left-most vowel

b) Parameter : A number

Output : The number with its digits in the reverse order

**//p3a.html**

<!DOCTYPE html PUBLIC "\_//w3c//DTD//XHTML 1.1//EN""http://www.w3.org/TR/xhtml11/DTD/xhtml.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<script type="text/javascript">

function disp(str)

{

var alphaexp=/^[A-Za-z]+$/;

if(!str.value.match(alphaexp))

{

alert("input should have only aphabets");

return false;

}

sml=31;

text=str.value.toLowerCase();

var ia=text.indexOf("a");

if(sml>ia&&ia>=0)

sml=ia;

var ie=text.indexOf("e");

if(sml>ie&&ie>=0)

sml=ie;

var ii=text.indexOf("i");

if(sml>ii&&ii>=0)

sml=ii;

var io=text.indexOf("o");

if(sml>io&&io>=0)

sml=io;

var iu=text.indexOf("u");

if(sml>iu&&iu>=0)

sml=iu;

if(sml==31)

alert("no vowels found");

else

alert("the position of the left most vowel is "+(sml+1));

}

</script>

</head>

<body>

<form>

enter a string<input type="text" name="string"size="30"maxlength="30"/>

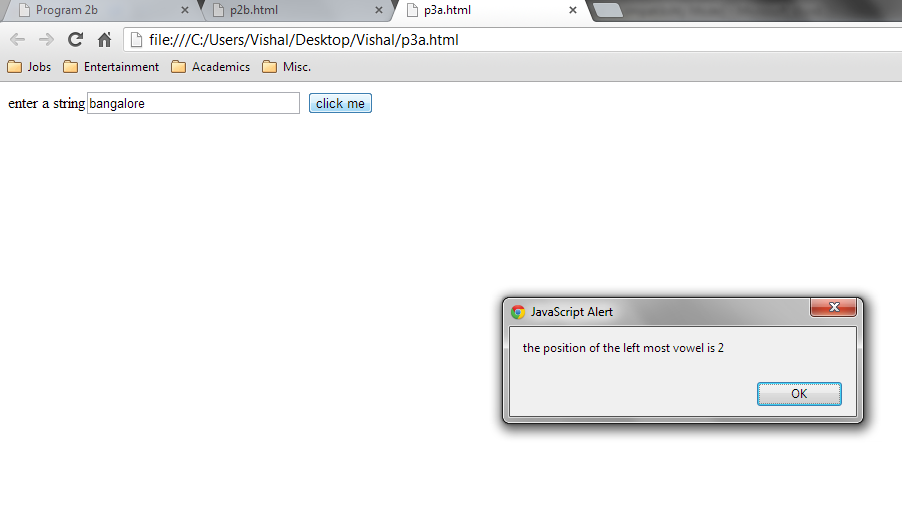
<input type="button"value="click me"onclick="disp(string)"/>

</form>

</body>

</html>

**OUTPUT:**



**//p3b.html**

<!DOCTYPE html PUBLIC "\_//w3c//DTD//XHTML 1.1//EN""http://www.w3.org/TR/xhtml11/DTD/xhtml.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<script type="text/javascript">

function disp(num)

{

var alphaexp=/^[0-9]+$/;

if(!num.value.match(alphaexp))

{

alert("input must have only numbers");

return false;

}

var rn=0,n=Number(num.value);

while(n!=0)

{

r=n%10;

n=Math.floor(n/10);

rn=rn\*10+r;

}

alert("the " +num.value+" in reverse is "+rn);

}

</script>

</head>

<body>

<form>

enter the number<input type="text" name="number"/>

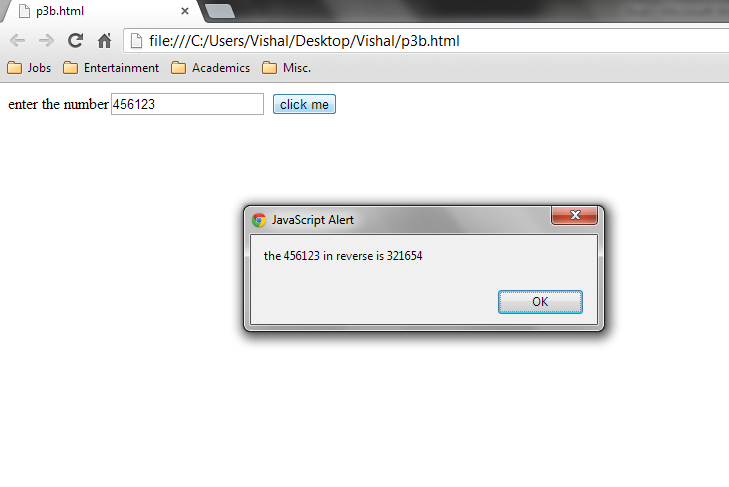
<input type="button"value="click me"onclick="disp(number)"/>

</form>

</body>

</html>

**OUTPUT:**



**PROGRAM 4**

a) Develop and demonstrate, using Javascript script, a XHTML document that collects the USN ( the valid format is: A digit from 1 to 4 followed by two upper-case characters followed by two digits followed by two upper-case characters followed by three digits; no embedded spaces allowed) of the user. Event handler must be included for the form element that collects this information to validate the input. Messages in the alert windows must be produced when errors are detected.

b) Modify the above program to get the current semester also (restricted to be a number from 1 to 8)

**//p4a.html**

<? xml version="1.0" encoding="utf-8" ?>

<!DOCTYPE html PUBLIC "\_//w3c//DTD//XHTML 1.1//EN""http://www.w3.org/TR/xhtml11/DTD/xhtml.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<script type="text/javascript">

function formValidator()

{

var usn=document.getElementById('req1');

if(isCorrect(usn))

{

return true;

}

return false;

}

function isCorrect(elem1)

{

var alphaexp1 = /[1-4][A-Za-z][A-Za-z][0-9][0-9][A-Za-z][A-Za-z][0-9][0-9][0-9]$/;

if(elem1.value.length==0)

{

alert("please enter the input");

elem1.focus();

return false;

}

else if(!elem1.value.match(alphaexp1))

{

alert("enter the input in DAADAADDD format");

elem1.focus();

return false;

}

else

{

alert("USN is correct");

return true;

}

}

</script>

</head>

<body>

<form onsubmit="return formValidator( )">

enter the usn<input type="text"id='req1'/>

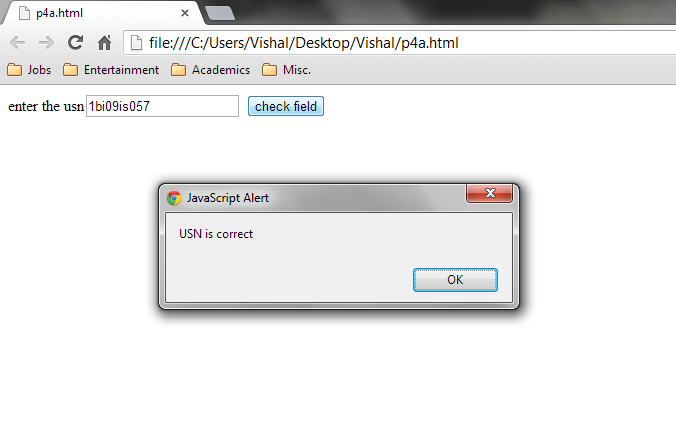
<input type="submit"value="check field"/>

</form>

</body>

</html>

**OUTPUT:**



**//p4b.html**

<!DOCTYPE html PUBLIC "\_//w3c//DTD//XHTML 1.1//EN""http://www.w3.org/TR/xhtml11/DTD/xhtml.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<script type="text/javascript">

function formValidator()

{

var usn=document.getElementById('req1');

var sem=document.getElementById('req2');

if(isCorrect(usn))

{

if(isPerfect(sem))

{

return true;

}

return false;

}

}

function isPerfect(elem2)

{var alphaexp2=/^[1-8]$/;

if(elem2.value.length==0)

{

alert("please enter the input");

elem2.focus();

return false;

}

else if(!elem2.value.match(alphaexp2))

{

alert("the sem should have a value in the range 1-8");

elem2.focus();

return false;

}

else

{ alert("SEMESTER is correct");

return true;

}

}

function isCorrect(elem1)

{

var alphaexp1=/[1-4][A-Za-z][A-Za-z][0-9][0-9][A-Za-z][A-Za-z][0-9][0-9][0-9]$/;

if(elem1.value.length==0)

{

alert("please enter the input");

elem1.focus();

return false;

}

else if(!elem1.value.match(alphaexp1))

{

alert("enter the input in DAADAADDD format");

elem1.focus();

return false;

}

else

{alert("USN is correct");

return true;

}

}

</script>

</head>

<body>

<form onsubmit="return formValidator()">

enter the usn<input type="text"id="req1"/><br/>enter the semester<input type="text"id="req2"/><br/>

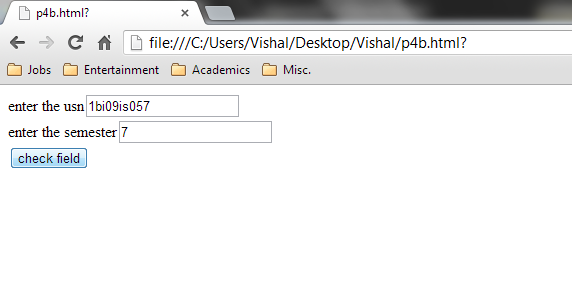
<input type="submit"value="check field"/>

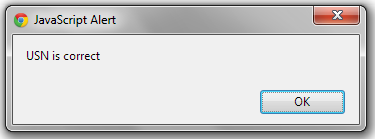
</form>

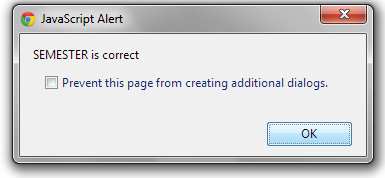
</body>

</html>

**OUTPUT:**







**PROGRAM 5**

a) Develop and demonstrate, using Javascript script, a XHTML document that contains three short paragraphs of text, stacked on top of each other, with only enough of each showing so that the mouse cursor can be placed over some part of them. When the cursor is placed over the exposed part of any paragraph, it should rise to the top to become completely visible.

b) Modify the above document so that when a paragraph is moved from the top stacking position, it returns to its original position rather than to the bottom.

**//5a.html**

<html>

<head>

<title>Stack</title>

<script type="text/javascript" >

var top="a1";

<!-- This function is such that, each time a particular paragraph is clicked, that paragraph’s zindex is increased to 10, thereby making it rise to the top and become completely visible -- >

function toTop(newTop)

{

domTop = document.getElementById(top).style;

domNew = document.getElementById(newTop).style;

domTop.zIndex="0";

domNew.zIndex="10";

top=newTop;

}

<!--We are making a simple style sheet of each paragraph, initializing all the zindex to 0, assigning different colors and relative positions to each of them -- >

</script>

<style type="text/css">

.box1 { position:absolute; top:0; left:0; z-index:0; background-color:#FF0000 }

.box2 { position:absolute; top:50px; left:100px; z-index:0; background-color:#0000FF}

.box3 { position:absolute; top:100px; left:200px; z-index:0; background-color:#FFFF00}

</style>

</head>

<!--The body contains the 3 paragraphs, everytime the mouse is placed over a particular paragraph, the “toTop()” function is called and that para rises to the top -- >

<body>

<p>

<p class="box1" id="a1" onmouseover="toTop('a1')">

11111111111111111111111111111111111111111111111<br />

11111111111111111111111111111111111111111111111<br />

11111111111111111111111111111111111111111111111<br />

11111111111111111111111111111111111111111111111<br />

11111111111111111111111111111111111111111111111<br />

11111111111111111111111111111111111111111111111<br />

</p>

<p class="box2" id="a2" onmouseover="toTop('a2')">

22222222222222222222222222222222222222222222222<br />

22222222222222222222222222222222222222222222222<br />

22222222222222222222222222222222222222222222222<br />

22222222222222222222222222222222222222222222222<br />

22222222222222222222222222222222222222222222222<br />

22222222222222222222222222222222222222222222222<br />

</p>

<p class="box3" id="a3" onmouseover="toTop('a3')" >

33333333333333333333333333333333333333333333333<br />

33333333333333333333333333333333333333333333333<br />

33333333333333333333333333333333333333333333333<br />

33333333333333333333333333333333333333333333333<br />

33333333333333333333333333333333333333333333333<br />

33333333333333333333333333333333333333333333333<br />

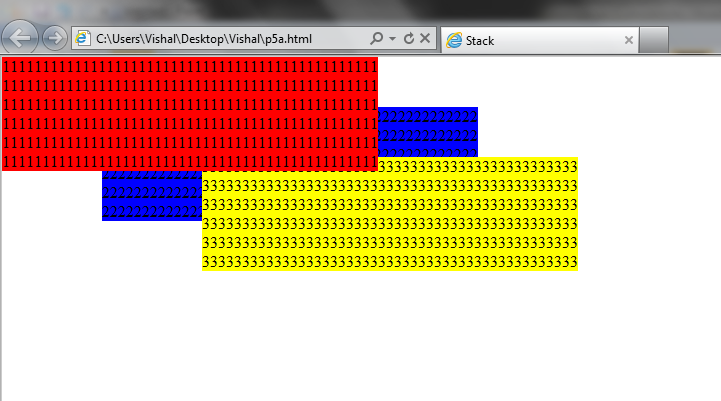
</p>

</p>

</body>

</html>

**OUTPUT:**



**//p5b.html**

<html>

<head>

<title>Stack</title>

<script type="text/javascript" >

var top="a1";

function toTop(newTop)

{

domTop = document.getElementById(top).style;

domNew = document.getElementById(newTop).style;

domNew.zIndex="10";

}

function toReg()

{

domNew = document.getElementById('a1').style;

domNew.zIndex="0";

domNew = document.getElementById('a2').style;

domNew.zIndex="1";

domNew = document.getElementById('a3').style;

domNew.zIndex="2";

}

</script>

<style type="text/css">

.box1 { position:absolute; top:0; left:0; z-index:0; background-color:#FF0000 }

.box2 { position:absolute; top:50px; left:100px; z-index:0; background-color:#0000FF}

.box3 { position:absolute; top:100px; left:200px; z-index:0; background-color:#FFFF00}

</style>

</head>

<body>

<p>

<p class="box1" id="a1" onmouseover="toTop('a1')" onmouseout="toReg()">

11111111111111111111111111111111111111111111111<br />

11111111111111111111111111111111111111111111111<br />

11111111111111111111111111111111111111111111111<br />

11111111111111111111111111111111111111111111111<br />

11111111111111111111111111111111111111111111111<br />

11111111111111111111111111111111111111111111111<br />

</p>

<p class="box2" id="a2" onmouseover="toTop('a2')" onmouseout="toReg()">

22222222222222222222222222222222222222222222222<br />

22222222222222222222222222222222222222222222222<br />

22222222222222222222222222222222222222222222222<br />

22222222222222222222222222222222222222222222222<br />

22222222222222222222222222222222222222222222222<br />

22222222222222222222222222222222222222222222222<br />

</p>

<p class="box3" id="a3" onmouseover="toTop('a3')" onmouseout="toReg()">

33333333333333333333333333333333333333333333333<br />

33333333333333333333333333333333333333333333333<br />

33333333333333333333333333333333333333333333333<br />

33333333333333333333333333333333333333333333333<br />

33333333333333333333333333333333333333333333333<br />

33333333333333333333333333333333333333333333333<br />

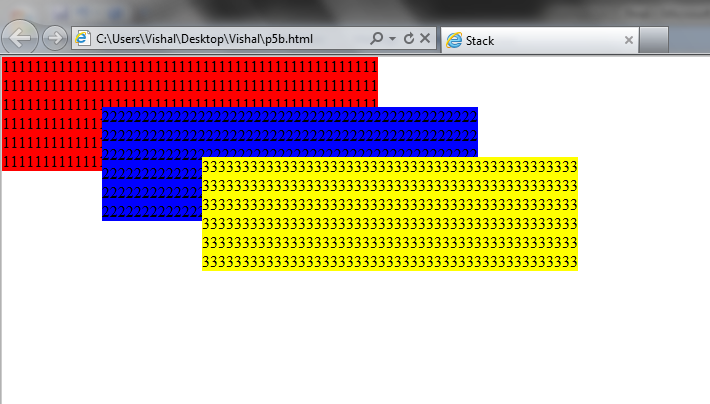
</p>

</p>

</body>

</html>

**OUTPUT:**



**PROGRAM 6**

a) Design an XML document to store information about a student in an engineering college affiliated to VTU. The information must include USN, Name, Name of the College, Brach, Year of Joining, and e-mail id. Make up sample data for 3 students. Create a CSS style sheet and use it to display the document.

b) Create an XSLT style sheet for one student element of the above document and use it to create a display of that element.

**//6a.xml**

<?xml version = "1.0" ?>

<?xml-stylesheet type = "text/xsl" href = "6a.xsl" ?> <students>

<vtu>

<usn>1BI07IS402</usn>

<name>Keerthi Kumar</name> <college>BIT</college>

<branch>ISE</branch>

<yoj>2006</yoj>

<email>keerthikumar@yahoo.co.in</email>

</vtu>

<vtu>

<usn>1BI07IS401</usn>

<name>Dheeraj Kumar</name> <college>BIT</college>

<branch>ISE</branch>

<yoj>2006</yoj>

<email>dheeraj@gmail.com</email>

</vtu>

<vtu>

<usn>1BI07IS404</usn>

<name>Puneeth</name>

<college>BIT</college>

<branch>ISE</branch>

<yoj>2006</yoj>

<email>puneeth@yahoo.com</email>

</vtu>

</students>

**//6a.xsl**

<?xml version = "1.0" ?>

<xsl:stylesheet version = "1.0"

xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"

xmlns = "http://www.w3.org/1999/xhtml" >

<xsl:template match = "students">

<h2> VTU STUDENT'S DESCRIPTIONS </h2> <xsl:for-each select ="vtu">

<span style = "font-style: italic; color: blue;">USN: </span>

<xsl:value-of select = "usn" /> <br />

<span style = "font-style: italic; color: blue;">NAME: </span>

<xsl:value-of select = "name" /> <br />

<span style = "font-style: italic; color: blue;">COLLEGE: </span>

<xsl:value-of select = "college" /> <br />

<span style = "font-style: italic; color: blue;">BRANCH: </span>

<xsl:value-of select = "branch" /> <br />

<span style = "font-style: italic; color: blue;">YEAR OF JOINING: </span>

<xsl:value-of select = "yoj" /> <br />

<span style = "font-style: italic; color: blue;">E-MAIL: </span>

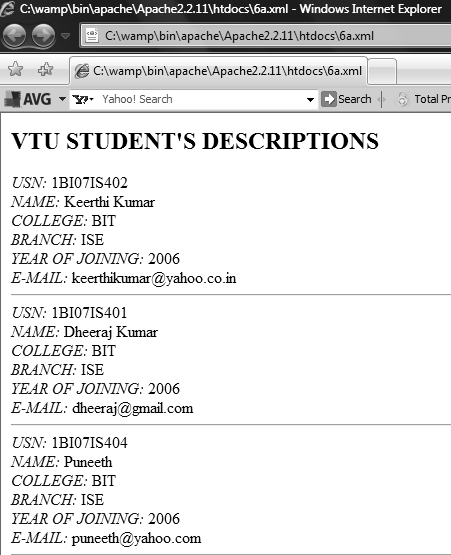
<xsl:value-of select = "email" /> <br /><hr />

</xsl:for-each>

</xsl:template>

</xsl:stylesheet>

**OUTPUT:**



**//6b.xml**

<?xml version = "1.0" ?>

<?xml-stylesheet type = "text/xsl" href = "6b.xsl" ?>

<vtu>

<usn>1BI07IS402</usn>

<name>Keerthi Kumar</name> <college>BIT</college>

<branch>ISE</branch>

<yoj>2006</yoj>

<email>keerthikumar@yahoo.co.in</email>

</vtu>

**//6b.xsl**

<?xml version = "1.0" ?>

<xsl:stylesheet version = "1.0"

xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"

xmlns = "http://www.w3.org/1999/xhtml" >

<xsl:template match = "vtu">

<html><head><title> STYLE SHEET FOR 6b.xml</title></head>

<body>

<h2> VTU STUDENT'S DESCRIPTIONS </h2>

<span style = "font-style: italic; color: blue;">USN: </span>

<xsl:value-of select = "usn" /> <br />

<span style = "font-style: italic; color: blue;">NAME: </span>

<xsl:value-of select = "name" /> <br />

<span style = "font-style: italic; color: blue;">COLLEGE: </span>

<xsl:value-of select = "college" /> <br />

<span style = "font-style: italic; color: blue;">BRANCH: </span>

<xsl:value-of select = "branch" /> <br />

<span style = "font-style: italic; color: blue;">YEAR OF JOINING: </span>

<xsl:value-of select = "yoj" /> <br />

<span style = "font-style: italic; color: blue;">E-MAIL: </span>

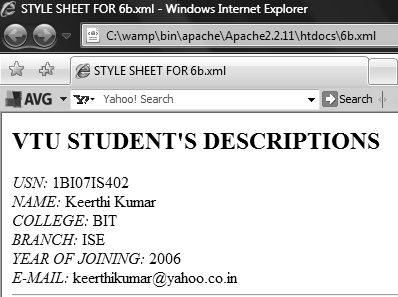
<xsl:value-of select = "email" /> <br /><hr />

</body></html>

</xsl:template>

</xsl:stylesheet>

**OUTPUT:**



**PROGRAM 7**

a) Write a Perl program to display various Server Information like Server Name, Server Software, Server protocol, CGI Revision etc.

b) Write a Perl program to accept UNIX command from a HTML form and to display the output of the command executed.

**//7a.pl**

#! C:/Perl/bin/perl -w

use CGI':standard';

print "content-type:text/html","\n\n";

print "<html>\n";

print "<head><title> About this server </title> </head>\n";

print "<body><h1> About this server </h1>","\n";

print "<hr>";

print "Server name:",$ENV{'SERVER\_NAME'},"<br>";

print "Running on port :",$ENV{'SERVER\_PORT'},"<br>";

print "Server Software :",$ENV{'SERVER\_SOFTWARE'},"<br>";

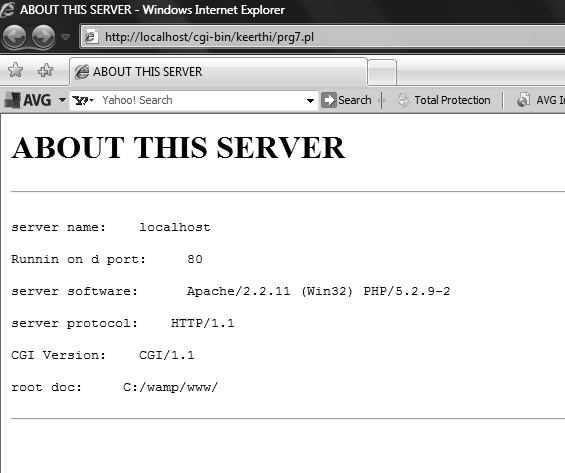
print "CGI-Revision :",$ENV{'GATEWAY\_INTERFACE'},"<br>";

print "<hr>\n";

print "</body></html>\n";

exit(0);

**OUTPUT:**



**//7b.pl**

#!C:/perl/bin/perl -w

use CGI':standard';

print "content-type: text/html \n\n";

$c=param('com');

system($c);

exit(0);

**//7b.html**

<html>

<body>

<form action="http://localhost/cgi-bin/vishal/7b.pl">

<input type="text" name="com">

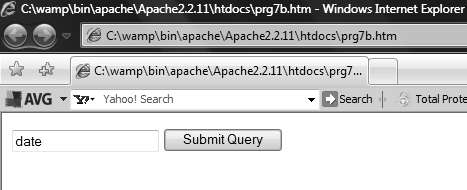
<input type="submit" name="submit">

</form>

</body>

</html>

**OUTPUT:**



**PROGRAM 8**

a) Write a Perl program to accept the User Name and display a greeting message randomly chosen from a list of 4 greeting messages.

b) Write a Perl program to keep track of the number of visitors visiting the web page and to display this count of visitors, with proper headings.

**//8a.pl**

#!c:/perl/bin/perl -w

use CGI ':standard';

use CGI::carp qw(WarningsToBrowser);

@coins=("welcome to web programming lab","have a nice day","hey","dear");

$range=4;

#>>>pick a random number in the range

$random\_number=int(rand($range));

if(param)

{

print header();

#>>>set the title in the title bar, the background color and the text color.

print start\_html(-title=>"username",-bgcolor=>"pink",-text=>"blue");

#>>>initialize the varible cmd to store the user entered input

$cmd=param("name");

#>>>print the data along with the randomly picked phrase in the array coins.

print b("hello $cmd,$coins[$random\_number]"),br();

print start\_form();

print submit(-value=>"back");

print end-form();

print end\_html();

}

else

{

print header();

print start\_html(-title=>"enter user name",-bgcolor=>"red",-text=>"black");

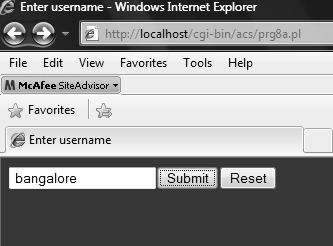
print start\_form(),textfield(-name=>"name",-value=>" "),submit(-name=>"submit",-value=>"submit"),reset();

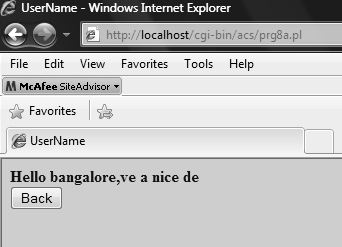
print end\_form();

print end\_html();

}

**OUTPUT:**





**//8b.pl**

#!c:/perl/bin/perl -w

use CGI ':standard';

use CGI::carp qw(WarningsToBrowser);

print header();

#>>>Set the Title for the title bar and background color and text colour.

print start\_html(-title=>"webpage counter",-bgcolor=>"pink",-text=>"blue");

open(FILE,'<count.txt');

#>>>Use a variable count to keep count of the number of times the page was opened.

$count=<FILE>;

close(FILE);

#increment the count whenever the page(file)is opened and closed.

$count++;

open(FILE,'>count.txt');

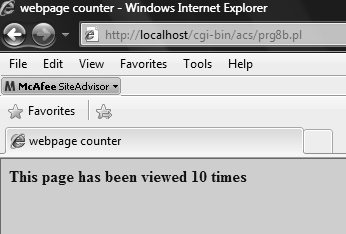
print FILE "$count";

print b("this page has been viewed $count times");

close(FILE);

print end\_html();

**OUTPUT:**



**PROGRAM 9**

Write a Perl program to display a digital clock which displays the current time of the server.

#!c:/perl/bin/perl -w # specifies where the perl interpreter is

use CGI ':standard'; #to use satndard module available in CGI.pm to generate html tags.

print "Refresh: 1\n"; #refresh the page after 1 sec.

print "Content-Type: text/html\n\n"; #setting content of the server response to html doc # uses CGI.pm to print the title, setting background colour

print start\_html(-title=>"Program 9",-bgcolor=>"Black",-text=>"White"); ($s,$m,$h)=localtime(time);

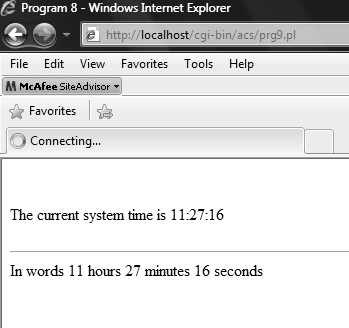
#to get the current time

print br,br,"The current system time is $h:$m:$s"; #equivalent to<br /><br /> of xhtml and print the time.

print br,br,hr,"In words $h hours $m minutes $s seconds";

print end\_html; #ends the HTML document by printing the </BODY> and </HTML> tags.

**OUTPUT:**



**PROGRAM 10**

Write a Perl 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.

**//10.pl**

#!c:/perl/bin/perl -w

print "Content-type: text/html\n\n";

print "<HTML><TITLE>Result of the insert operation </TITLE>"; use CGI ':standard';

use DBI;

#to connect to data base interface with database name “vishal”….login ID as “root”..and password as #“ise123”…

$dbh=DBI->connect("DBI:mysql:vishal","root","ise123");

$name=param("name"); #to store value from HTML page text box “name”

$age=param("age"); #to store age from HTML page into a variable

$qh=$dbh->prepare("insert into stud values('$name','$age')"); # to insert the value into the table #created in the datbase

$qh->execute(); #interact with the data base and execute the query following it.

$qh=$dbh->prepare("Select \* from stud"); #to retrieve all data from table name “stud” $qh->execute();

print "<table border size=1><tr><th>Name</th><th>Age</th></tr>";

while ( ($name,$age)=$qh->fetchrow())

#to display data retrieve from database to the HTML page

{

print "<tr><td>$name</td><td>$age</td></tr>";

}

print "</table>";

$qh->finish(); #To inform that operation is finished

$dbh->disconnect(); #close the connection to the database print"</HTML>";

**//10.html**

<html>

<body>

<form action="http://localhost/cgi-bin/vishal/10.pl">

Name : <input type="text" name="name"> <br>

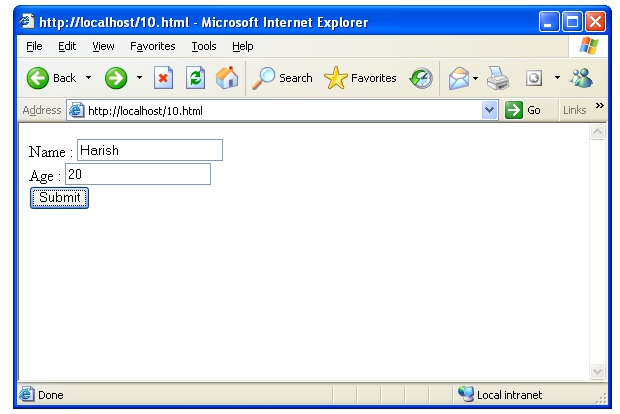
Age :<input type="text" name="age"> <br>

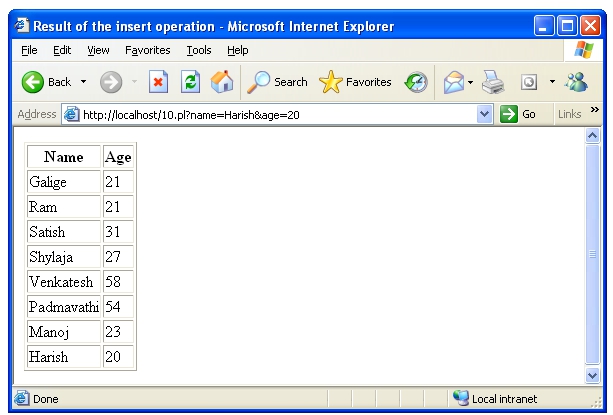
<input type="submit" value="Submit">

</form>

</html>

**OUTPUT:**





**PROGRAM 11**

Write a PHP program to store current date-time in a COOKIE and display the ‘Last visited on’ date-time on the web page upon reopening of the same page.

<?php //start of the program

date\_default\_timeZone\_set('Asia/Calcutta'); //to set the time zone

$inTwoMonths=60\*24\*60+time(); //calculate 60 days in the future to set the lifetime of the cookie.

setcookie('lastVisit',date("G:i-m/d/y"),$inTwoMonths);

//to create cookie with name lastVisit whose value is the date and time returned by the date()

if(isset($\_COOKIE['lastVisit'])) //checking if the cookie came with a request.

{

$visit=$\_COOKIE['lastVisit']; //visit saves the cookie value.

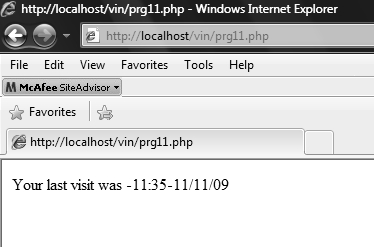
echo "Your last visit was -".$visit; //to print the time and date of last visit.

}

else

echo "you ve got some stale cookies!"; ?> //end of PHP program

**OUTPUT:**



**PROGRAM 12**

Write a PHP program to store page views count in SESSION, to increment the count on each refresh, and to show the count on web page.

<?php

session\_start(); // a session ID is created and recorded.

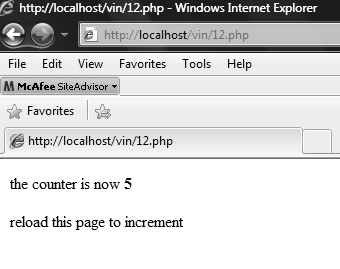
session\_register("count"); //session key value pairs are created or changed by assignments to the \_SESSION array. $\_SESSION["count"]++; //each time the page is loaded, the count is incremented.

echo "<p>the counter is now <b>$\_SESSION[count]</b></p>".

"<p>reload this page to increment</p>";

?>

**OUTPUT:**



**PROGRAM 13**

Create a XHTML form with Name, Address Line 1, Address Line 2, and E-mail text fields. On submitting, store the values in MySQL table. Retrieve and display the data based on Name.

**//13a.php (Data Insertion)**

<?php

$dbh=mysql\_connect('localhost','root','ise123')or

die(mysql\_error());

mysql\_select\_db('vishal')or die(mysql\_error());

if(isset($\_POST['name']))

$nme=$\_POST['name'];

$ad1=$\_POST['add1'];

$ad2=$\_POST['add2'];

$em1=$\_POST['email'];

if($nme!=""&&$ad1!="")

{

$query="INSERT INTO contact VALUES('$nme','$ad1','$ad2','$em1')";

$result=mysql\_query($query) or die(mysql\_error());

}

else

{

echo "one of the field is empty";

}

mysql\_close($dbh);

echo "Data Inserted <br><hr>";

?>

**//13b.php (Data Retrieval)**

<?php

$link=mysql\_connect("localhost","root","ise123");

mysql\_select\_db("vishal");

$n=$\_POST["name"];

print "Entered Name is $n \n";

$var=mysql\_query("SELECT \* FROM contact WHERE name like '%$n%'");

echo"<table border size=1>";

echo"<tr> <th>Name</th> <th>Addr1</th>

<th>Addr2</th><th>Email</th></tr>";

while ($arr=mysql\_fetch\_row($var))

{echo "<tr><td>$arr[0]</td><td>$arr[1]</td><td>$arr[2]</td>

<td>$arr[3]</td></tr>";}

echo"</table>";

mysql\_free\_result($var);

mysql\_close($link);

?>

**//13.html (Common interface for Data Insertion and Retrieval)**

<html>

<head><title>pg13</title></head>

<form action="http://localhost/vishal/13a.php" method="post">

<p>

name:<input type=text name="name" value=""><br>

address1:<input type=text name="add1" value=""><br>

address2:<input type=text name="add2" value=""><br>

email:<input type=text name="email" value=""><br>

<input type=submit>

</p>

</form>

<hr>

<form action="http://localhost/vishal/13b.php" method="post">

Enter Name of the contact to be searched <input type="text" name="name">

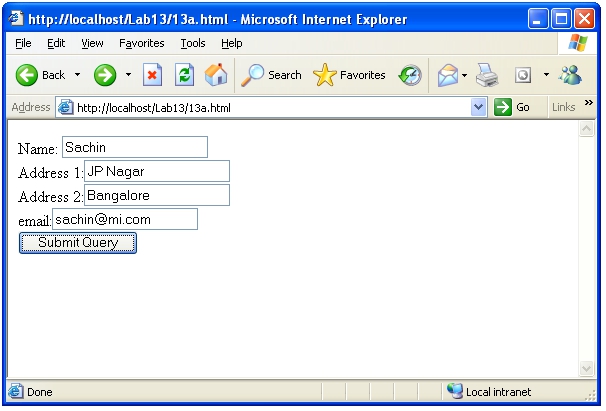
<input type=submit>

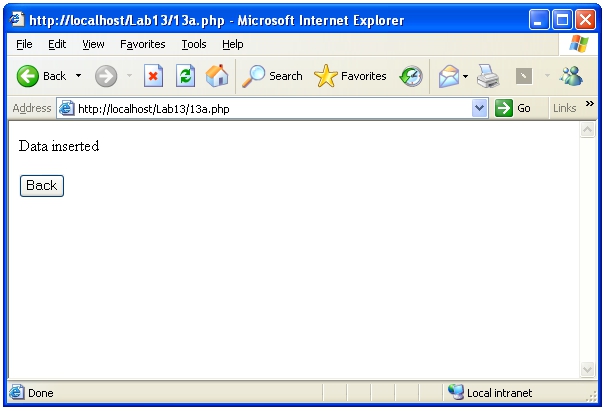
</form>

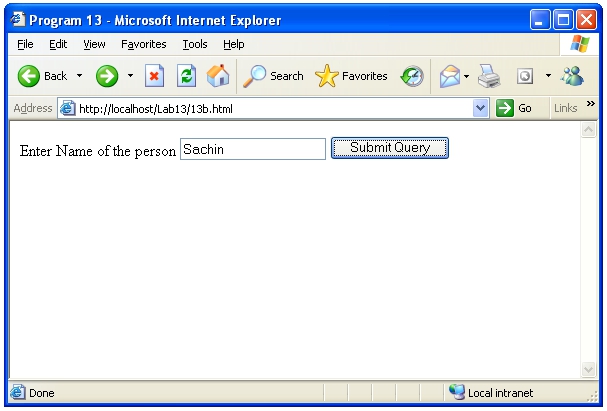
</body>

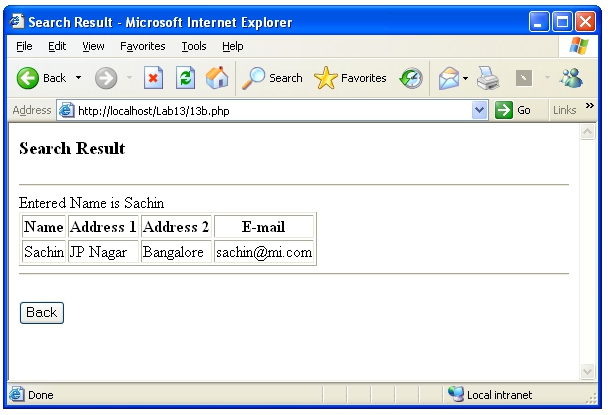
</html>

**OUTPUT:**









**PROGRAM 14**

Using PHP and MySQL, develop a program to accept book information viz. Accession number, title, authors, edition and publisher from a web page and store the information in a database and to search for a book with the title specified by the user and to display the search results with proper headings.

**//14a.php (Data Insertion)**

<?php

$dbh=mysql\_connect('localhost','root','ise123')or

die(mysql\_error());

mysql\_select\_db('vishal')or die(mysql\_error());

if(isset($\_POST['title']))

$acc=$\_POST['acc'];

$nme=$\_POST['title'];

$ad1=$\_POST['add1'];

$ad2=$\_POST['add2'];

$em1=$\_POST['email'];

if($nme!=""&&$ad1!="")

{

$query="INSERT INTO book VALUES('$acc','$nme','$ad1','$ad2','$em1')";

$result=mysql\_query($query) or die(mysql\_error());

}

else

{

echo "one of the field is empty";

}

mysql\_close($dbh);

echo "Data Inserted <br><hr>";

?>

**//14b.php (Data Retrieval)**

<?php

<?php

$link=mysql\_connect("localhost","root","ise123");

mysql\_select\_db("vishal");

$n=$\_POST["title"];

print "Entered Name is $n \n";

$var=mysql\_query("SELECT \* FROM book WHERE title ='$n'");

echo"<table border size=1>";

echo"<tr><th>Acc No.</th> <th>Title</th> <th>author</th>

<th>Edition</th><th>Publisher</th></tr>";

while ($arr=mysql\_fetch\_row($var))

{echo "<tr><td>$arr[0]</td><td>$arr[1]</td><td>$arr[2]</td>

<td>$arr[3]</td><td>$arr[4]</td> </tr>";}

echo"</table>";

mysql\_free\_result($var);

mysql\_close($link);

?>

**//14.html (Common interface for Data Insertion and Retrieval)**

<html>

<head><title>pg13</title></head>

<form action="http://localhost/vishal/14a.php" method="post">

<p>

acc no:<input type=text name="acc" value=""><br>

title:<input type=text name="title" value=""><br>

author:<input type=text name="add1" value=""><br>

edition:<input type=text name="add2" value=""><br>

publisher:<input type=text name="email" value=""><br>

<input type=submit>

</p>

</form>

<hr>

<form action="http://localhost/vishal/14b.php" method="post">

Enter Name of the book to be searched <input type="text" name="name">

<input type=submit>

</form>

</body>

</html>

**OUTPUT:**

