Q.2a. Write the HTML code for the following table:

Firstname	Lastname	Points
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

Ans2a.

```
DO<!CTYPE html>
<html>

<head>
<style>
table,th,td
{
border:1px solid black;
border-collapse:collapse;
}
th,td
{
padding:5px;
}
</style>
</head>

<body>
<table style="width:300px">
<tr>
  <th>Firstname</th>
  <th>Lastname</th>
  <th>Points</th>
  </tr>
<tr>
<tr>
  <td>Jill</td>
  <td>Smith</td>
  <td>50</td>
  </tr>
```

```
        <tr>
          <td>Eve</td>
          <td>Jackson</td>
          <td>94</td>
        </tr>
        <tr>
  <td>John</td>
  <td>Doe</td>
  <td>80</td>
</tr>
</table>
</body>
</html>
```

b.Write the HTML code to accomplish the web page:

(i) Insert the frame extending 300 pixels across the page from left side.
(ii) Insert an image onto a page using good.gif as image and having "welcome" as the ALT text.

Ans2b.
(i)

```
< FRAMESET COLS = " 300 , * " >
. . .
< /FRAMESET >
```

(i i)

```
<IMG SRC = " good.gif " ALT = " Welcome"
```

c.Explain using suitable examples, the Syntactic differences between HTML and XHTML?

Ans2c. (Explain any three syntactic differences between HTML and XHTML,give example and syntax of tags)
Case sensitivity: In HTML, tag and attribute names are case insensitive meaning that <FORM>, <form>,and <Form> are equivalent. In XHTML, all tag and attribute names must be in lowercase. Closing tags: In HTML, closing tag may be omitted if the processing agent can infer heir presence. For example, in HTML, paragraph elements often do not have closing tags. For example <p> During Spring, flowers are born. … <p> During Fall, flowers die….. HTML documents are case insensitive whereas XHTML documents have to be in lowercase Closing tags may be omitted in HTML but not in XHTML where all elements must have closing tags except for content tags where it is not a must <input type= ―text‖ name=―address‖ /> Quoted attribute values :all attribute values must be quoted whether it is

numeric or character based Explicit attribute values: In HTML some attributes are implicit Quoted attribute values: In HTML, attribute values must be quoted only if there are embedded special characters or white space characters. Explicit attribute values: In HTML, some attribute values are implicit, that is, they need not be explicitly stated. For example, if the border attribute appears in a <table> tag without a value, it specifies a default width border on the table. For example: <table border>. id and name attributes. HTML markup often uses the name attribute for elements. Element nesting: Although HTML has rules against improper nesting of elements, they are not enforced.

Ans3a.

```
 <style type="text/css">

  body {

   text-align:center;

height:100%;

font-size:62.5%;

font-family: Helvetica, arial, sans-serif;

color: #000;
```

```
      background: #eceae1 url(/images/bg_body.png) repeat-x 0 0;

    margin: 0;

    padding: 0;

      }



  a {

      color:#007CA5;

      text-decoration:none;

      outline: none;
```

```
   }

a:visited { color:#666; }

a:active  { color:#999; }

img       { border: 0px; }

h1, .h1 { font-size: 2.4em;}

h2, .h2 { font-size: 2em;} /* article headline 20pt */

h3, .h3 { font-size: 1.8em;} /*  should be 18pt */
```

```
  </style>
```

b. What do you mean by paged media? How CSS2 styles for paged media? Explain using appropriate syntax/code.

Ans3b. Paged media differ from continuous media in that the content of the document is split into one or more discrete pages. Paged media includes paper, transparencies, pages that are displayed on computer screens, etc.

The CSS2 standard introduces some basic pagination control features that let authors help the browser figure out how to best print their documents.
The CSS2 page model specifies how a document is formatted within a rectangular area -- the page box -- that has a finite width and height. These features fall into two groups:

- CSS2 features that define a particular page layout.
- CSS2 features that control the pagination of a document.

The CSS2 defines a "page box", a box of finite dimensions in which content is rendered. The page box is a rectangular region that contains two areas:

- **The page area:** The page area includes the boxes laid out on that page. The edges of the page area act as the initial containing block for layout that occurs between page breaks.
- **The margin area:** which surrounds the page area.

You can specify the dimensions, orientation, margins, etc. of a page box within an @page rule. The dimensions of the page box are set with the 'size' property. The dimensions of the page area are the dimensions of the page box minus the margin area.
For example, the following @page rule sets the page box size to 8.5 x 11 inches and creates '2cm' margin on all sides between the page box edge and the page area:

```
<style tyle="text/css">
<!--
@page { size:8.5in 11in; margin: 2cm }
-->
</style>
```

Setting Page Size:
The *size* property specifies the size and orientation of a page box. There are four values which can be used for page size:

auto: The page box will be set to the size and orientation of the target sheet.

- **landscape:** Overrides the target's orientation. The page box is the same size as the target, and the longer sides are horizontal. **portrait:** Overrides the target's orientation. The page box is the same size as the target, and the shorter sides are horizontal.
- **length:** Length values for the 'size' property create an absolute page box. If only one length value is specified, it sets both the width and height of the page box. Percentage values are not allowed for the 'size' property.

In the following example, the outer edges of the page box will align with the target. The percentage value on the 'margin' property is relative to the target size so if the target sheet dimensions are 21.0cm x 29.7cm (i.e., A4), the margins are 2.10cm and 2.97cm.

```
<style tyle="text/css">
<!--
@page {
  size: auto;   /* auto is the initial value */
  margin: 10%;
}
-->
</style>
```

Q.4 a. What are advantages of dynamic HTML (DHTML)? Briefly describe various components of DHTML.

Ans4a.

(1) DHTML makes documents dynamic. Dynamic documents:

o Allow the designer to control how the HTML displays Web pages'content.

o React and change with the actions of the visitor.

o Can exactly position any element in the window, and change that position after the document has loaded.

o Can hide and show content as needed.

(2) DHTML allows any HTML element (any object on the screen that can be controlled independently using JavaScript) in Internet Explorer to be manipulated at any time, turning plain HTML into dynamic HTML

(3) With DHTML, changes occur entirely on the client-side (on the user's browser).

(4) Using DHTML gives the author more control over how the page is formatted and how content is positioned on the page.

Components of DHTML

Dynamic HTML includes the following components:

o Conventional HTML

o Scripts – Small programs designed to manipulate Web pages.

o Document Object Model (DOM) – The road map through which you can locate any element in an HTML document and use a scripting language, such as JavaScript, to change the element's properties.

o Absolute Positioning – The elements on the page are placed in a fixed location, as opposed to relative positioning, in which an element's location is relative to particular elements on the page.

o Multimedia filters – Multimedia features that create visual effects for text, images, and other objects, without imposing long download times on the user.

b.What are the governing rules for forming name of JavaScript variable? JavaScript has

four types of data, discuss about them.

Syntax try { statements; } catch (error) { statements; } finally { statements; }
If anything goes wrong in the statements that are inside the try block's statements then the statements in the catch block will be executed and the error will be passed in the error variable. The finally block is optional and, if present, is always executed last, regardless if there was an error caught or not.

Q.5a. Write a JavaScript function that checks when submitted if the form data has the general syntax of an email.

Ans5a.

```
<!DOCTYPE html>
<html>
<head>
<script>
function validateForm()
{
var x=document.forms["myForm"]["email"].value;
var atpos=x.indexOf("@");
var dotpos=x.lastIndexOf(".");
if (atpos<1 || dotpos<atpos+2 || dotpos+2>=x.length)
  {
  alert("Not a valid e-mail address");
  return false;
  }
}
</script>
</head>

<body>
<form name="myForm" action="demo_form.asp" onsubmit="return validateForm();" method="post">
Email: <input type="text" name="email">
<input type="submit" value="Submit">
</form>
</body>
</html>
```

b.Write a script to place your logo on your Web site in such a way that it floats on the page as the user scrolls up and down.

Ans5b. floatingLogo.js

```
var x1 = 11;   // change the # on the left to adjust the X co-ordinate
var y1 = 130;  // change the # on the left to adjust the Y co-ordinate
```

```
(document.getElementById && !document.all) ? dom = true : dom = false;

function typeStart() {
  if (dom) {
    document.write('<div id="logoBox" style="position:absolute; left:' + (window.innerWidth-x1) + 'px;
                visibility:visible">') }
  if (document.all) {
    document.write('<div id="logoBox" style="position:absolute; left:' +
                (document.documentElement.clientWidth-x1) + 'px; visibility:visible">') }
 }

function typeEnd() {
  if (document.all || dom) { document.write('</div>') }
 }

function placeIt() {
  if (dom) {document.getElementById("logoBox").style.top = window.pageYOffset + y1 + "px";
                document.getElementById("logoBox").style.left = window.pageXOffset + x1 + "px";}
  if (document.all) {document.all["logoBox"].style.top = document.documentElement.scrollTop + y1 +
                "px"; document.all["logoBox"].style.left = document.documentElement.scrollLeft + x1 +
                "px";}
  window.setTimeout("placeIt()", 10);
 }

function addLoadEvent(func) {
  var oldonload = window.onload;
  if (typeof window.onload != 'function') {
    window.onload = func;
  } else {
    window.onload = function() {
      if (oldonload) {
        oldonload();
      }
      func();
    }
  }
}

addLoadEvent(function() {
  placeIt();
});
```

Head

Paste this code into the HEAD section of your HTML document.

<script type="text/javascript" src="floatingLogo.js"></script>

Body
Paste this code into the BODY section of your HTML document

```
<script>typeStart()</script>
 <ahref="http://www.javascriptsource.com"><img
        src="http://www.javascriptsource.com/img/headerlogo.gif"        style="border:
        none;"></a>
<script>typeEnd()</script>
```

Q.6a. Is there a difference between an undefined array variable and an array variable containing the empty list? Write a PERL program that count the number of words In the standard input file.

Ans6a. No. By default, all array variables contain the empty list. Note, however, that the empty list is not the same as a list containing the null string: `@array = ("");`
This list contains one element, which happens to be a null string.

```
 #!/usr/local/bin/perl
 $wordcount = 0;
 $line = <STDIN>;
 while ($line ne "") {
        chop ($line);
        @array = split(/ /, $line);
        $wordcount += @array;
        $line = <STDIN>;
}
print ("Total number of words: $wordcount\n");
```

b. What are nested subroutines? Can it is used in PERL? Write an example program that demonstrates use of nested subroutines in PERL.

Ans6b. Subroutines that are called by other subroutines are known as *nested subroutines*. In Perl, you can call subroutines from other subroutines. To call a subroutine from another subroutine, use the same subroutine-invocation syntax you've been using all along.
An example of a nested subroutine.

```
 #!/usr/local/bin/perl
 ($wordcount, $charcount) = &getcounts(3);
 print ("Totals for three lines: ");
 print ("$wordcount words, $charcount characters\n");
  sub getcounts {
  my ($numlines) = @_;
         my ($charpattern, $wordpattern);
        my ($charcount, $wordcount);
        my ($line, $linecount);
        my (@retval);
        $charpattern = "";
        $wordpattern = "\\s+";
        $linecount = $charcount = $wordcount = 0;

        while (1) {
            $line = <STDIN>;
              last if ($line eq "");
              $linecount++;
              $charcount += &count($line, $charpattern);
              $line =~ s/^\s+|\s+$//g;
              $wordcount += &count($line, $wordpattern);
              last if ($linecount == $numlines);
        };
        @retval = ($wordcount, $charcount);
}

sub count {
        my ($line, $pattern) = @_;
       my ($count);
        if ($pattern eq "") {
              @items = split (//, $line);
        } else {
              @items = split (/$pattern/, $line);
        }
        $count = @items;
}
```

Q.7a. In addition to the required name=value pair, each cookie has several optional attributes. Explain any four of these.

Ans7a. In addition to the required name=value pair, each cookie has several optional attributes:

1. an expiration time

This is a time/date string (in a special GMT format) that indicates when a cookie expires. The cookie will be saved and returned to your script until this expiration date is reached if the user exits the browser and restarts it. If an expiration date isn't specified, the cookie will remain active until the user quits the browser.

2. a domain

This is a partial or complete domain name for which the cookie is valid. The browser will return the cookie to any host that matches the partial domain name. For example, if you specify a domain name of ".leo.com", then the browser will return the cookie to Web servers running on any of the machines "www.leo.com", "www2.leo.com", "feckless.leo.com", etc. Domain names must contain at least two periods to prevent attempts to match on top level domains like ".edu". If no domain is specified, then the browser will only return the cookie to servers on the host the cookie originated from.

3. a path

If you provide a cookie path attribute, the browser will check it against your script's URL before returning the cookie. For example, if you specify the path "/cgi-bin", then the cookie will be returned to each of the scripts "/cgi-bin/tally.pl", "/cgi-bin/order.pl", and "/cgi-bin/customer_service/complain.pl", but not to the script "/cgi-private/site_admin.pl". By default, path is set to "/", which causes the cookie to be sent to any CGI script on your site.

4. a "secure" flag

If the "secure" attribute is set, the cookie will only be sent to your script if the CGI request is occurring on a secure channel, such as SSL.

b. For what purpose HTML::Mason is used? Explain how to install HTML::Mason. Write a code in HTML::Mason to print the date today

Ans7b.

With HTML::Mason, the Perl script is embedded in the HTML code. This means that the HTML skeleton is coded and logic code is written only in portions of the script that needs it.

Installing HTML::Mason

HTML::Mason works with Apache 1.3 with mod_perl in native mode. On Apache 2.0 and mod_perl 2.0, HTML::Mason runs in CGI mode.

Installation instructions for HTML::Mason on for Apache 1.3 and mod_perl 1.0 is straightforward. Before installing HTML::Mason, make sure that mod_perl is already installed on Apache. Download the module from CPAN or from the HTML::Mason website.

Do a :

```
perl Makefile
make && make test && make install
```

Once installed, you will need to configure the Apache server. Edit the httpd.conf file and make sure you include this:

```
PerlModule HTML::Mason::ApacheHandler
<Location /usr/local/apache/htdocs/mason>
```

```
SetHandler perl-script
PerlHandler HTML::Mason::ApacheHandler
</Location>
```

You can also use the Directory or Files directive to make HTML::Mason work in specific directory or files. Installation instructions for HTML::Mason on Apache 2.0 and mod_perl 2.0 are available at: http://beaucox.com/mason/mason-with-apmp...

Sample HTML::Mason Code

If you want to generate a code to show the date today, you can do this:

```
<%perl>
use Date::Calc;
my @today  = Date::Calc->Today();
my $str = "$today[0]-$today[1]-$today[2]";
</%perl>

<html>
<body>
Today is <%$str %>
</body>
</html>
```

Q.8a. Explain how can you track users on the internet using PHP.

Ans8a. You can track users on the internet by setting cookies. They've been used for this probably ever since someone invented them.

We can simply set a cookie to the user when he opens our page and when he comes back we can see that he has the cookie we set, provided he hasn't deleted it or such.

To know the exact time they visited the last time, for showing new things since their last visit for example, the cookie's value can be set to the time when the user is visiting. This way when he returns we can look at the cookie and see the time when he was visiting earlier.

```
<?php
//First we need to check if the user has visited before
//Let's use 'visittime' as the name for our cookie

if(isset($_COOKIE['visittime']))
{
  echo 'Welcome back!<br>';

  //Calculate how long is since the users last visit
  //We can do this simply by subtracting the last visit
  //time from the current time
  $timeElapsed = time() - $_COOKIE['visittime'];

  echo 'It has been ' . $timeElapsed . ' seconds since your last visit.';
}
else
{
  echo 'I see you are new here';
}

//Set the current time as a cookie and make it expire much later.
$nextYear = time() + (60 * 60 * 24 * 365);
setcookie('visittime', time(), $nextYear);
?>
```

b. Using suitable PHP code, explain how can you create a table in a database.

```
<?php

/*** mysql hostname ***/
$hostname = 'localhost';

/*** mysql username ***/
$username = 'username';

/*** mysql password ***/
$password = 'password';

/*** connect to the database ***/
$link = @mysql_connect($hostname, $username, $password);

/*** check if the link is a valid resource ***/
if(is_resource($link))
    {
    /*** if we are successful ***/
    echo 'Connected successfully<br />';

    /*** select the database we wish to use ***/
    if(mysql_select_db("periodic_table", $link) === TRUE)
        {
        /*** sql to create a new table ***/
        $sql = "CREATE TABLE elements (
        atomicnumber tinyint(3) NOT NULL default '0',
        latin varchar(20) NOT NULL default '',
        english varchar(20) NOT NULL default '',
        abbr char(3) NOT NULL default '',
        PRIMARY KEY  (atomicnumber)
        )";

        /*** run the sql query ***/
        if(mysql_query($sql, $link))
            {
            echo 'New table created successfully<br />';
            }
        else
            {
            echo 'Unable to create table: <br />' . $sql .'<br />' . mysql_error();
            }
        }
    /*** if we are unable to select the database show an error ***/
    else
        {
        echo 'Unable to select database';
        }
    /*** close the connection ***/
    mysql_close($link);
    }
else
    {
    /*** if we fail to connect ***/
    echo 'Unable to connect';
    }
?>
```

Q.9a. What do you mean by DTD? How elements and attributes are specified in DTD? Explain with examples.

Ans9a.
The purpose of a DTD (Document Type Definition) is to define the legal building blocks of an XML document.
A DTD defines the document structure with a list of legal elements and attributes.

In a DTD, elements are declared with an ELEMENT declaration.

Declaring Elements
In a DTD, XML elements are declared with an element declaration with the following syntax:

```
<!ELEMENT element-name category>
or
```

```
<!ELEMENT element-name (element-content)>
```

Empty Elements

Empty elements are declared with the category keyword EMPTY:

```
<!ELEMENT element-name EMPTY>

Example:

<!ELEMENT br EMPTY>

XML example:

<br />
```

Declaring Attributes

An attribute declaration has the following syntax:

```
<!ATTLIST element-name attribute-name attribute-type attribute-value>

DTD example:

<!ATTLIST payment type CDATA "check">

XML example:

<payment type="check" />
```

The **attribute-type** can be one of the following:

Type	Description
CDATA	The value is character data
(*en1*|*en2*|..)	The value must be one from an enumerated list
ID	The value is a unique id
IDREF	The value is the id of another element
IDREFS	The value is a list of other ids
NMTOKEN	The value is a valid XML name
NMTOKENS	The value is a list of valid XML names
ENTITY	The value is an entity

ENTITIES	The value is a list of entities
NOTATION	The value is a name of a notation
xml:	The value is a predefined xml value

The **attribute-value** can be one of the following:

Value	Explanation
value	The default value of the attribute
#REQUIRED	The attribute is required
#IMPLIED	The attribute is optional
#FIXED *value*	The attribute value is fixed

b. What is XML::Parser? Give an overview of the same. Write a PERL program that demonstrate any one of XML::Parser's capabilities.

Ans9b. *XML::Parser is a Perl module that acts as an interface to expat, James Clark's XML parser. A prototype was originally created by Larry Wall, and Clark Cooper has continued the development of this useful tool*
Most Perl applications in need of an XML parser will likely fall into one of two types. The first type of application will process specific applications of XML, for example RDF or MathML. For these, a subclass of `XML::parser` will need to be written in order to provide a tool conceptually closer to the job at hand. The second type of application will operate on any conforming XML document in order to find or filter out pieces of the document, or to discover things about its structure.
Like James Clark's `expat` library, upon which it's built, `XML::Parser` is an event-based parser. Prior to parsing the document, an application registers various event handlers with the parser. Then, as the document is parsed, the handlers are called when the relevant parts are recognized.
Most utilities need only register 3 handlers: `start`, `end`, and `character` handlers. The start handler is called when an XML start tag is recognized; the end handler is called on recognition of an end tag; and the character handler is called for non-markup content inside an element.

Text Book

1. **Web Programming – Building Internet Applications, Chris Bates, Third Edition, Wiley Student Edition, 2006**