Programming the Web (10CS73)

December – 2017

1. a. Describe with figure how domain name conversion happens on the web. (5 M)

The IP addresses are numbers. Hence, it would be difficult for the users to remember IP address. To solve this problem, text based names were introduced. These are technically known as ***domain name system (DNS).*** These names begin with the names of the host machine, followed by progressively larger enclosing collection of machines, called ***domains.*** There may be two, three or more domain names. DNS is of the form **hostname.domainName.domainName** . Example: **rnsit.ac.in** The steps for conversion from DNS to IP: The DNS has to be converted to IP address before destination is reached. This conversion is needed because computer understands only numbers. The conversion is done with the help of *name server*. As soon as domain name is provided, it will be sent across the internet to contact name servers. This name server is responsible for converting domain name to IP If one of the *name servers* is not able to convert DNS to IP, it contacts other name server. This process continues until IP address is generated. Once the IP address is generated, the host can be accessed. The hostname and all domain names form what is known as FULLY QUALIFIED DOMAIN NAME. This is as shown below:

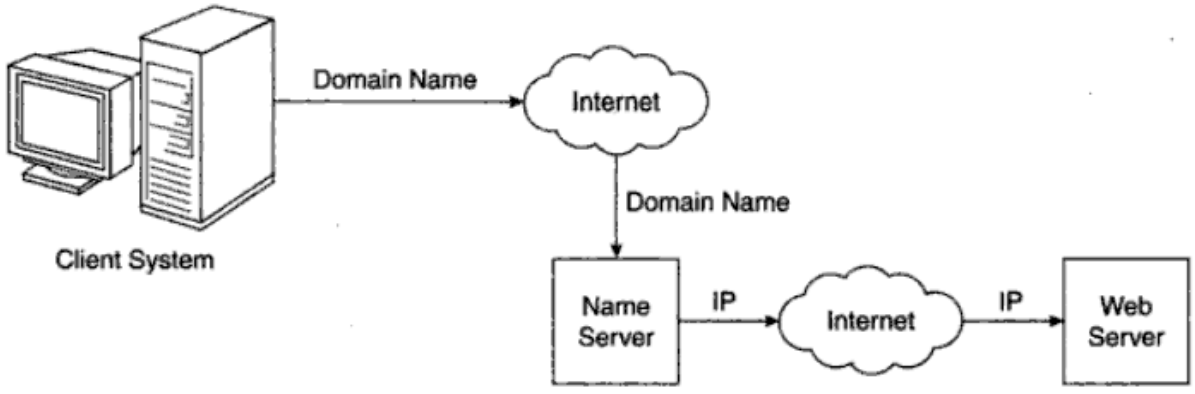


Figure 1.1 Domain name conversion

b. Explain the syntactic differences between HTML and XHTML. (5M)

PARAMETERS	HTML	XHTML
Case Sensitivity	Tags and attributes names are case insensitive	Tags and attributes names must be in lowercase
Closing tags	Closing tags may be omitted	All elements must have

		closing tag
Quoted attribute values	Special characters are quoted. Numeric values are rarely quoted.	All attribute values must be quoted including numbers
Explicit attribute values	Some attribute values are implicit. For example: <table border>. A default value for border is assumed	All attribute values must be explicitly stated
***id* and *name* attributes**	Both *id* and *name* attributes are encouraged	Use of *id* is encouraged and use of *name* is discouraged
Element nesting	Rules against improper nesting of elements (for example: a form element cannot contain another form element) are not enforced.	All nesting rules are strictly enforced

c) Explain the following tags with examples: (10 M)

i) img ii)a iii) pre iv)sub v) meta

i) img

image tag is to add an image to the document.

ii) a

anchor tag is used to add hypertext or hyper image or any sort of hyper documents.

click me

iii) <pre>

Sometimes it is desirable to preserve the white space in text—that is, to prevent the browser from eliminating multiple spaces and ignoring embedded line breaks. This can be specified with the <pre> tag. <html> <head> <title> Pre Tag </title> </head> <body> <pre> My Name is Chetan I am from CSE Department SJBIT, Bangalore </pre> </body> </html>

iv)sub

This tag is to add subscript to the data.

H₂O

v) <meta>

The meta element (for search engines) Used to provide additional information about a document, with attributes, not content. <meta id=‖a‖></meta>

2 a). Explain the different levels of stylesheet. (6M)

LEVELS OF STYLE SHEETS The three levels of style sheets, in order from lowest level to highest level, are inline, document level, and external.

Inline style sheets apply to the content of a single XHTML element.

Document-level style sheets apply to the whole body of a document.

External style sheets can apply to the bodies of any number of documents.

Inline style sheets have precedence over document style sheets, which have precedence over external style sheets.

Inline style specifications appear within the opening tag and apply only to the content of that tag.

Document-level style specifications appear in the document head section and apply to the entire body of the document

External style sheets stored separately and are referenced in all documents that use them

External style sheets are written as text files with the MIME type text/css.

They can be stored on any computer on the Web. The browser fetches external style sheets just as it fetches documents.

The <link> tag is used to specify external style sheets. Within <link>, the rel attribute is used to specify the relationship of the linked-to document to the document in which the link appears.

The href attribute of <link> is used to specify the URL of the style sheet document.

EXAMPLE WHICH USES EXTERNAL STYLE SHEET

```
<html>
        <head> <title>Sample CSS</title>
        <link rel = "stylesheet" type = "text/css" href = "Style1.css" />
       </head>
        <h1>Puneeth Rajkumar</h1>
</html>
```

Style1.css

```
h1 { font-family: 'Lucida Handwriting'; font-size: 50pt; color: Red; }
```

EXAMPLE WHICH USES DOCUMENT LEVEL STYLE SHEET

```
<html>
        <head> <title>Sample CSS</title>
```

```
<style type = "text/css">
h1 { font-family: 'Lucida Handwriting'; font-size: 50pt; color: Red; }
</style>
</head>
<h1>Puneeth Rajkumar</h1>
</html>
```

EXAMPLE WHICH USES INLINE STYLE SHEET

```
<html>
        <head> <title>Sample CSS</title> </head>
        <h1 style ="font-family: 'Lucida Handwriting'; font-size: 50pt; color: Red;">
        Puneeth Rajkumar
        </h1>
</html>
```

b)Create test and validate an XHTML document that defines a table with columns for states, state bird state flower and state trees. There must be atleast three states as rows in the table. Include cellpadding and cellspacing attributes. (7M)

```
<?xml version = "1.0" encoding = "utf-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
 "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!-- table.html
    An example of a simple table
    -->
<html xmlns = "http://www.w3.org/1999/xhtml">
 <head> <title> A simple table </title>
 </head>
 <body>
  <table border = "border" cellpadding="4px" cellspacing="4px">
   <caption>State Emblems</caption>
   <tr>
    <th> State</th>
    <th> Bird</th>
    <th> Tree </th>
    <th> Flower </th>
   </tr>
   <tr>
```

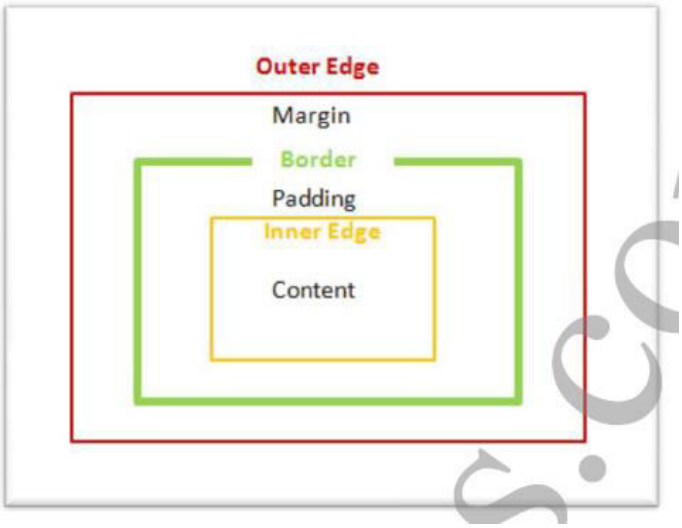
```
      <th> Karnataka </th>
      <td>Indian roller</td>
      <td> Sandal </td>
      <td> Lotus </td>
    </tr>
    <tr>
      <th> Tamil Nadu </th>
      <td> Emerald dove </td>
      <td> Palmyra palm</td>
      <td> Kandhal</td>
    </tr>
    <tr>
      <th> Kerala </th>
      <td> Dinner </td>
      <td> Coconut</td>
      <td> Golden shower tree </td>
    </tr>
  </table>
 </body>
</html>
```

State Emblems

State	Bird	Tree	Flower
Karnataka	Indian roller	Sandal	Lotus
Tamil Nadu	Emerald dove	Palmyra palm	Kandhal
Kerala	Dinner	Coconut	Golden shower tree

c)With a neat figure of box model, explain borders, margin and padding. (7 M)

THE BOX MODEL



- On a given web page or a document, all the elements can have borders.
- The borders have various styles, color and width.
- The amount of space between the content of the element and its border is known as *padding*.
- The space between border and adjacent element is known as *margin.*

Borders:

Border-style

It can be dotted, dashed, double
Border-top-style
Border-bottom-style
Border-left-style
Border-right-style

Border-width

It can be thin, medium, thick or any length value
Border-top-width
Border-bottom-width
Border-left-width
Border-right-width

Border-color

Border-top-color
Border-bottom-color
Border-left-color
Border-right-color

<html> <head> <title> Table with border effects </title> <style type = "text/css"> table { border-width:thick; border-top-color:red; border-left-color:orange; border-bottom-color:violet; border-right-color:green; border-top-style:dashed;

Margins and Padding: The margin properties are named margin, which applies to all four sides of an element: margin-left, margin-right, margin-top, and margin-bottom. The padding properties are named padding, which applies to all four sides: padding-left, padding-right, padding-top, and padding-bottom.

<html>

<head>

<title> Margins and Padding </title>

<style type = "text/css">

p.one { margin:0.1in; padding:0.5in; background-color:#FF33FF; border-style:dotted; }

p.two { margin:0.5in; padding:0.1in; background-color:#00FF33; border-style:dashed; }

p.three {margin:0.3in; background-color:#FFFF00; }

p.four { padding:0.3in; background-color:#FF9900; }

</style>

</head>

<body>

 Puneeth Rajkumar is the Power Star of Sandalwood
 [margin=0.1in, padding=0.5in]

 Puneeth Rajkumar is the Power Star of Sandalwood
 [margin=0.5in, padding=0.1in]

 Puneeth Rajkumar is the Power Star of Sandalwood
 [margin=0.3in, no padding, no border]

 Puneeth Rajkumar is the Power Star of Sandalwood
 [no margin, padding=0.3in, no border] </body> </html>

3 a. Explain the javascript string property and methods with example. (6 M)

The String object includes one property, length, and a large collection of methods. The number of characters in a string is stored in the length property as follows: var str = “George”; var len = str.length; //now, len=6

Method	Parameters	Result
charAt	A number	Returns the character in the String object that is at the specified position
indexOf	One-character string	Returns the position in the String object of the parameter
substring	Two numbers	Returns the substring of the String object from the first parameter position to the second
toLowerCase	None	Converts any uppercase letters in the string to lowercase
toUpperCase	None	Converts any lowercase letters in the string to uppercase

Consider, **var str = “George”;**

Now, **str.charAt(2) is „o“ str.indexOf(„r“) is 3 str.substring(2, 4) is „org“ str.toLowerCase() is „george“**

b.Write a javascript to check validity of a phone number. (5 M)

```
<?xml version = "1.0" encoding = "utf-8" ?>
<!DOCTYPE html PUBLIC "-//w3c//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!-- validator.html
     A document for validator.js
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head>
    <title> Illustrate form input validation> </title>
    <script type = "text/javascript"  src = "validator.js" >
    </script>
  </head>
  <body>
    <h3> Customer Information </h3>
    <form action = "">
      <p>
        <label>
          <input type = "text"  id = "phone" />
```

```
          Phone number (ddd-ddd-dddd)
        </label>
        <br /><br />

        <input type = "reset"  id = "reset" />
           <input type = "submit"  id = "submit" />
      </p>
    </form>
    <script type = "text/javascript">
      <!--
// Set form element object properties to their
// corresponding event handler functions
        document.getElementById("custName").onchange = chkName;
        document.getElementById("phone").onchange = chkPhone;
      // -->
    </script>
  </body>
</html>
// validator.js
//   An example of input validation using the change and submit
//   events
     // The event handler function for the name text box
// The event handler function for the phone number text box
function chkPhone() {
  var myPhone = document.getElementById("phone");
// Test the format of the input phone number
  var pos = myPhone.value.search(/^\d{3}-\d{3}-\d{4}$/);
  if (pos != 0) {
    alert("The phone number you entered (" + myPhone.value +
          ") is not in the correct form. \n" +
          "The correct form is: ddd-ddd-dddd \n" +
          "Please go back and fix your phone number");
    myPhone.focus();
    myPhone.select();
    return false;
  } else
    return true;
}
```

c) Explain any six javascript array methods.(6 M)

Array METHODS

Array objects have a collection of useful methods, most of which are described in this section.

- The **join** method converts all of the elements of an array to strings and catenates them into a single string. If no parameter is provided to join, the values in the new string are separated by commas. If a string parameter is provided, it is used as the element separator. Consider the following example:

```
var names = new Array["Mary", "Murray", "Murphy", "Max"];
...
var name_string = names.join(" : ");
```

The value of `name_string` is now "Mary : Murray : Murphy : Max".

- The **reverse** method reverses the order of the elements of the Array object through which it is called.
- The **sort** method coerces the elements of the array to become strings if they are not already strings and sorts them alphabetically
- The **concat** method catenates its actual parameters to the end of the Array object on which it is called.

- The **slice** method does for arrays what the substring method does for strings, returning the part of the Array object specified by its parameters, which are used as subscripts. The array returned has the elements of the Array object through which it is called, from the first parameter up to, but not including, the second parameter.
- The value of list2 is now [4, 6]. If slice is given just one parameter, the array that is returned has all of the elements of the object, starting with the specified index.
- When the **toString** method is called through an Array object, each of the elements of the object is converted (if necessary) to a string. These strings are catenated, separated by commas. So, for Array objects, the toString method behaves much like join.
- The **push, pop, unshift,** and **shift** methods of Array allow the easy implementation of stacks and queues in arrays. The pop and push methods respectively remove and add an element to the high end of an array, as in the following code:

- The shift and unshift methods respectively remove and add an element to the beginning of an array. **var deer = list.shift(); // deer is now "Dasher" list.unshift("Dasher"); // This puts "Dasher" back on list**

4 a. What are the three ways of accessing XHTML document elements in javascript? (8 M)

The elements of an XHTML document have corresponding objects that are visible to an embedded JavaScript script. There are several ways the object associated with an XHTML form element can be addressed in JavaScript. The original (DOM 0) way is to use the forms and elements arrays of the Document object, which is referenced through the document property of the Window object. Example:

The DOM address of the button in this example, using the forms and elements arrays, is as follows: **var dom = document.forms[0].elements[0];** The problem with this approach to element addressing is that the DOM address is defined by address elements that could change—namely, the forms and elements arrays.

The elements of an XHTML document have corresponding objects that are visible to an embedded JavaScript script. There are several ways the object associated with an XHTML form element can be addressed in JavaScript. The original (DOM 0) way is to use the forms and elements arrays of the Document object, which is referenced through the document property of the Window object.

Example:

The DOM address of the button in this example, using the forms and elements arrays, is as follows: **var dom = document.forms[0].elements[0];** The problem with this approach to element addressing is that the DOM address is defined by address elements that could change— namely, the forms and elements arrays. Another Approach:

Using the name attributes, the button's DOM address is as follows: **var dom = document.myForm.turnItOn;** One drawback of this approach is that the XHTML 1.1 standard does not allow the name attribute in the form element, even though the attribute is now valid for form elements. This is a validation problem, but it causes no difficulty for browsers.
Hence, alternatively we use, **var dom = document.getElementById("turnItOn");**

Buttons in a group of checkboxes often share the same name. The buttons in a radio button group always have the same name. In these cases, the names of the individual buttons obviously cannot be used in their DOM addresses.

The checked property of a checkbox object is set to true if the button is checked. For the preceding sample checkbox group, the following code would count the number of checkboxes that were checked:

b. Explain the following with an example. (12 M)

i) stacking of elements ii) Element visibility iii)Absolute positioning iv) Dynamic content.

STACKING ELEMENTS

- The top and left properties allow the placement of an element anywhere in the two dimensions of the display of a document.
- Although the display is restricted to two physical dimensions, the effect of a third dimension is possible through the simple concept of stacked elements, such as that used to stack windows in graphical user interfaces.
- Although multiple elements can occupy the same space in the document, one is considered to be on top and is displayed.
- The top element hides the parts of the lower elements on which it is superimposed.
- The placement of elements in this third dimension is controlled by the z-index attribute of the element.
- An element whose z-index is greater than that of an element in the same space will be displayed over the other element, effectively hiding the element with the smaller z-index value.
- The JavaScript style property associated with the z-index attribute is zIndex.

```
//stack.html <html >

 <head>

<title>Stacking Paragraphs</title>

<style type="text/css">

.para1 { border: solid thick #C0C0C0; padding: 1in; width:180px; background-color:#0000D0; color:white; position:absolute; top:70px; left:4in; z-index:1; }

 .para2 { border: solid thick #808000; padding: 1in; width:180px; background-color:red; color:white; position:absolute; top:105px; left:5in; z-index:2; }

.para3 { border: solid thick #00ffff; padding: 1in; width:180px; background-color:green; color:white; position:absolute; top:140px; left:6in; z-index:3; }

.display { font-size:25pt; color:blue; text-align:center; }

p:hover{background-color:rgb(250,200,150);font-size:25px;color:white;};
```

```
</style>

<script type="text/javascript">

 var stack1="stack1"; function move(curStack)

{

var oldStack=document.getElementById(stack1).style; oldStack.zIndex="0"; var
newStack=document.getElementById(curStack).style; newStack.zIndex="10";
stack1=document.getElementById(curStack).id;

 } </script>

 </head>

 <body>

 <h2 class="display">Stacking of Paragraphs on top of each other</h2>

<p class="para1" id="stack1" onmouseover="move('stack1')"> Dr R N Shetty </p> <p
class="para2" id="stack2" onmouseover="move('stack2')"> Dr H N ShivShankar </p> <p
class="para3" id="stack3" onmouseover="move('stack3')"> Dr M K Venkatesha </p>
</body>

 </html>
```

ELEMENT VISIBILITY

- Document elements can be specified to be visible or hidden with the value of their visibility property.
- The two possible values for visibility are, quite naturally, visible and hidden.
- The appearance or disappearance of an element can be controlled by the user through a widget.

//showHide.html

```
 <html>

<head>

 <title>Visibility Control</title>

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

</script>
```

```
</head>
 <body> <form action="">
<div id="rnsit" style="position:relative; visibility: visible;">
<img src = "CLASS PHOTO.jpg" alt = "RNSIT ISE 2009-2013" />
 </div>
<p><br><input type="button" value="Toggle Rnsit" onclick = "flipImag()" /></p>
 </form>
</body>
</html>
```

//showHide.js

```
function flipImag()
{
dom=document.getElementById("rnsit").style; if(dom.visibility == "visible") dom.visibility =
"hidden"; else dom.visibility = "visible";
 }
```

ABSOLUTE POSITIONING

- The absolute value is specified for position when the element is to be placed at a specific place in the document display without regard to the positions of other elements.
- One use of absolute positioning is to superimpose special text over a paragraph of ordinary text to create an effect similar to a watermark on paper.
- A larger italicized font, in a light-gray color and with space between the letters, could be used for the special text, allowing both the ordinary text and the special text to be legible.

//absPos.html

```
<html>
 <title>Absolute Positioning</title>
<style type = "text/css">
.regtext { font-family: Cambria; font-size:20pt; width: 900px; }
.abstext { position:absolute; top:25px; left:100px; font-family: jokerman; font-size:30pt; width:
500px; color:#0000cd; letter-spacing: 1em; }
</style>
</head>
```

<body>
 Kannadada Kotyadhipathi is a Kannada primetime quiz show hosted by the power star of Kannada cinema Mr. Puneet Rajkumar. This is the biggest game show ever on Kannada Television. This show will be aired on Suvarna TV. This show gives the common man an opportunity to win Rs 1 crore. Kannadada Kotyadipathi is a Kannada primetime quiz and human drama show hosted by matinee idol Puneeth Rajkumar on Suvarna TV. Contestants participate in a game that allows them to win up to Rs. 1 crore. Short-listed contestants play a 'Fastest Finger First' round to make it to the main game. From there on, they play rounds with increasing levels of difficulty, and winning higher amounts of money, culminating in the Rs. 1 crore prize. Contestants can stop at any time having viewed the next question. Or they can avail of a 'Lifeline' and play on. Welcome to the world of high stakes chills and thrills! Welcome to the world of the crorepati!
 POWER STAR PUNEETH RAJKUMAR
</body>
</html>

DYNAMIC CONTENT

- Assistance to a browser user filling out a form can be provided with an associated text area, often called a help box.
- The content of the help box can change, depending on the placement of the mouse cursor.
- When the cursor is placed over a particular input field, the help box can display advice on how the field is to be filled in.
- When the cursor is moved away from an input field, the help box content can be changed to simply indicate that assistance is available.
- When the mouse cursor is placed over an input field, the mouseover event is used to call a handler function that changes the help box content to the appropriate value
- The appropriate value is specified with a parameter sent to the handler function.
- The mouseout event is used to trigger the change of the content of the help box back to the "standard" value.

//dynValue.html

<html>

<head>

<title>Dynamic Values</title>

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

</head>

<body>

<form action="">

  Customer Information

<label>Name: <input type="text" onmouseover="messages(0)" onmouseout="messages(4)" /> </label>

<label>Email: <input type="text" onmouseover="messages(1)" onmouseout="messages(4)" /> </label>

 To create an account, provide the following information:

<label>User ID: <input type="text" onmouseover="messages(2)" onmouseout="messages(4)" /> </label>

<label>Password: <input type="text" onmouseover="messages(3)" onmouseout="messages(4)" /> </label>

<textarea id="adviceBox" rows="3" cols="50" style="position:absolute; left:250px; top:0px"> This box provides advice on filling out the form on this page. Put the mouse cursor over any input field to get advice. </textarea>

 <input type="submit" value="Submit" /> <input type="reset" value="Reset" /> </form> </body> </html> //dynValue.js var helpers = ["Your name must be in the form: \n first name, middle initial., last name", "Your email address must have the form: \ user@domain", "Your user ID must have at least six characters and it must include one digit", "This box provides advice on filling out the form on this page. Put the mouse cursor over any input field to get advice"]

function messages(adviceNumber) {
document.getElementById("adviceBox").value=helpers[adviceNumber]; }

5 a) Create a XML document for one student of VTU to illustrate formatting. Create XSLT style by child templates.

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet href="4b.xsl" type="text/xsl"?>
<vtu>
   <student-info> Student Information </student-info>
      <student>
      <name>anand</name>
```

```
      <usn>1CR12IS009</usn>
      <branch>ISE</branch>
      <college>CMRIT</college>
      <yoj>2012</yoj>
      <email>anand.r@cmrit.ac.in</email>
   </student>
</vtu>
```

XSLT styling

```
<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>
<body>
   <center><lable style="color:red;font-size:400%;font-style:italic;text-align:center;"> Student
Information</lable></center><br/>
      <xsl:for-each select="student">
            <lable style="color:green;font-size:300%;font-style:normal;">NAME:</lable>
            <span style="color:blue;font-size:300%;font-style:normal;">
            <xsl:value-of select="name"/>
            </span><br/>
            <lable style="color:black;font-size:250%;font-style:normal;">USN:</lable>
            <span style="color:red;font-size:250%;font-style:normal;">
            <xsl:value-of select="usn"/>
            </span><br/>
            <lable style="color:pink;font-size:200%;font-style:normal;">BRANCH:</lable>
            <span style="color:blue;font-size:200%;font-style:normal;">
            <xsl:value-of select="branch"/>
            </span><br/>
            <lable style="color:green;font-size:150%;font-style:normal;">COLLEGE:</lable>
            <span style="color:red;font-size:150%;font-style:normal;">
            <xsl:value-of select="college"/>
            </span><br/>
            <lable style="color:green;font-size:120%;font-style:normal;">YEAR:</lable>
            <span style="color:blue;font-size:120%;font-style:normal;">
            <xsl:value-of select="yoj"/>
            </span><br/>
            <lable style="color:blue;font-size:100%;font-style:normal;">EMAIL:</lable>
            <span style="color:green;font-size:100%;font-style:normal;">
            <xsl:value-of select="email"/>
```

```
            </span><br/>
        </xsl:for-each>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

b) Create XML document that list ads for used airplane. Create a DTD for the same document.

```
<!DOCTYPE XML_doc_root_name SYSTEM
“DTD_file_name”>
For examples,
<!DOCTYPE planes_for_sale SYSTEM
“planes.dtd”>
<?xml version = "1.0" encoding = "utf-8"?>
<!-- planes.xml - A document that lists ads for used airplanes -->
<!DOCTYPE planes_for_sale SYSTEM "planes.dtd">
<planes_for_sale>
<ad>
<year> 1977 </year>
<make> &c; </make>
<model> Skyhawk </model>
<color> Light blue and white </color>
<description> New paint, nearly new interior,
685 hours SMOH, full IFR King avionics </description>
<price> 23,495 </price>
<seller phone = "555-222-3333"> Skyway Aircraft </seller>
<location>
<city> Rapid City, </city>
<state> South Dakota </state>
</location>
</ad>
</planes_for_sale>
```

Sample DTD:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!-- planes.dtd - a document type definition for the planes.xml document, which specifies
a list of used airplanes for sale -->
```

```
<!ELEMENT planes_for_sale (ad+)>
<!ELEMENT ad (year, make, model, color, description, price?, seller, location)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT make (#PCDATA)>
<!ELEMENT model (#PCDATA)>
<!ELEMENT color (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT seller (#PCDATA)>
<!ELEMENT location (city, state)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ATTLIST seller phone CDATA #REQUIRED>
<!ATTLIST seller email CDATA #IMPLIED>
<!ENTITY c "Cessna">
<!ENTITY p "Piper">
<!ENTITY b "Beechcraft">
```

6 a) Explain the arrays and hashes in perl.(8 M)

1) An array is a variable that stores a list
   The name of an array variable begins with the character @
   An array variable may be assigned a literal list value
   - @a = (1, 2, ‘three’, ‘iv’);

   An array assignment creates a new array as a copy of the original
   - @b = @a;

Example.

```
@ages = (25, 30, 40);

@names = ("John Paul", "Lisa", "Kumar");

print "\$ages[0] = $ages[0]\n";

print "\$ages[1] = $ages[1]\n";

print "\$ages[2] = $ages[2]\n";

print "\$names[0] = $names[0]\n";
```

```
print "\$names[1] = $names[1]\n";

print "\$names[2] = $names[2]\n";
```

2) **Hash variables** are named beginning with the character %

If an array is assigned to a hash, the even index elements become keys and the odd index elements are the corresponding values

- Assigning an odd length array to a hash causes an error

Curly braces are used to 'subscript' a hash

- If %h is a hash, then the element corresponding to 'four' is referenced as $h{'four'}

Values can be assigned to a hash reference to insert a new key/value relation or to change the value related to a key

A key/value relation can be removed from a hash with the delete operator

The undef operator will delete all the contents of a hash

The exists operator checks if a key is related to any value in a hash

- Just check $h{'something'} doesn't work since the related value may be the empty string or 0, both of which count as boolean false

Ex: A hash variable embedded in a string is not interpolated

```
%countries=(India=>"Delhi", USA=>"Washington");
foreach $capital(keys %countries)
{
 print   "capital is" $countries{$capital} \n";
}
```

b) Explain session and cookies in perl. (6 M)

Session:

To generate a brand new session for a user

```
$session = new CGI::Session("driver:File", undef, {Directory=>"/tmp"});
```

Directory refers to a place where the session files and their locks will be stored in the form of separate files. When you generate the session object, as we did above, you will have:

1. Session ID generated and

2. Storage file associated with the id in the directory

GET and POST requests under HTTP can be used to carry form data from the browser to the server

- The data is formatted into a *query string*
- Each form of request includes the information in a different way
    - In a GET request, the query string is appended to the URL of the request, with a question mark used to separate it from the first part of the URL
    - In a POST request, the query string is sent as the data part of the request
- In both cases, the query string is formatted the same
- When the POST method is used, the query string can be read from standard input
    - The CONTENT_LENGTH environment variable tells how many characters can be read
- When The GET method is used, the query string is given by the value of the environment variable QUERY_STRING

This technique is built upon the aforementioned technologies plus a server-side storage device, which saves the state data for a particular session. Each session has a unique id associated with the data in the server. This id is also associated with the user agent in either the form of a cookie, a query_string parameter, a hidden field or all at the same time.

Advantages:

- We no longer need to depend on the User Agent constraints in cookie amounts and sizes
- Sensitive data like user's username, email address, preferences and such no longer need to be traveling across the network at each request (which is the case with query strings, cookies and hidden_fields). Only thing that travels across the network is the unique id generated for the session ("ID-1234", for instance), which should make no sense to bad guys whatsoever.
- User will not have sensitive data stored in his computer in an unsecured plain text format (which is a cookie file).
- It's possible to handle very big and even complex (in-memory) data structures transparently.

Cookies :

HTTP is a *stateless* protocol, that is, the server treats each request as completely separate from any other

This, however, makes some applications difficult

- A shopping cart is an object that must be maintained across numerous requests and responses

The mechanism of cookies can be used to help maintain state by storing some information on the browser system

A cookie is a key/value pair that is keyed to the domain of the server

- This key/value pair is sent along with any request made by the browser of the same server

A cookie has a lifetime which specifies a time at which the cookie is deleted from the browser. Cookies are only returned to the server that created them. Cookies can be used to determine usage patterns that might not otherwise be ascertained by a server. Browsers generally allow users to limit how cookies are used. Browsers usually allow users to remove all cookies currently stored by the browser

- Systems that depend on cookies will fail if the browser refuses to store them. The cookie function takes a hash with three keys for the name, value and expiration time of a cookie .The cookie value produced by this function must be passed to the header function using the –cookie key

  header(-cookie => $a_cookie)

- Calling the cookie function with no arguments produces a hash of all cookies from the current request. The day_cookie.pl example illustrates using a cookie to store the last time the page was visited.

c) **Write a note on CGI –PM module**.(6 M)

The Perl module CGI.pm provides numerous functions

1) print br;
puts the tag
 into the response

2) print h1("A Header")
puts
<h1>A Header</h1>
into the response

3) print textarea(-name => "Description", -rows => "2", -cols => "35");
produces this in the response
<textarea name="Description" rows="2" cols="35">
</textarea>

4)print a({-href => "fruit.html"},
Press here for fruit descriptions");
produces this in the response
 Press here for fruit descriptions

5)The head shortcut function provides a standard header

6) The start_html function provides the beginning part of an HTML document, through the <body> start tag

7 a) List and explain any 6 commonly used string functions in php. (7 M)

- Php provides many useful string functions
    - **strlen** gives the length of a string

      ```
      <?php
      echo strlen("Hello");
      ?>
      ```

    - **strcmp** compares two strings as strings

      ```
      <?php
      echo strcmp("Hello world!","Hello world!");
      ?>
      ```

    - **Chop** removes whitespace from the end of a string

      ```
      <?php
      $str = "Hello World!";
      echo $str . "<br>";
      echo chop($str,"World!");
      ?>
      ```

- **strchr** finds the first occurrence of a string inside another string

```
<?php
echo strchr("Hello world!","world");
?>
```

- **Strpos** returns the position of the first occurrence of a string inside another string

```
<?php
echo strpos("I love php, I love php too!","php");
?>
```

- **strrev** reverses the string

```
<?php
```

```
echo strrev("Hello World!");
?>
```

b) Write a note on php files. (7 M)

c) Explain session tracking in web applications. (6 M)

- The PHP function fopen is used to create a file handle for accessing a file given by name A second argument to fopen gives the mode of access

  The fopen function returns a file handle

- The file_exists function tests if a file, given by name, exists
- The function fclose closes a file handle
- The fread function reads a given number of bytes from a file given by a file handle
- The file function returns an array of lines from a file named as a parameter
- The file_get_contents method returns the content of a named file as a single string
- The fgetc function returns a single character
- The feof function returns TRUE if the last character read was the end of file marker, that is, the read was past the end of the file
- If a file handle is open to for writing or appending, then the fwrite function can be used to write bytes to the file
- The file_put_contents function writes a given string parameter to a named file, not a file handle

- Some applications need to keep track of a session. A session creates a file in a temporary directory on the server where registered session variables and their values are stored. This data will be available to all pages on the site during that visit.
- Sessions are represented internally in PHP with a session id
  - A session consists of key/value pairs
- A session can be initialized or retrieved by using the session_start function
  - This function retrieves $_SESSION, an array containing the key/value pairs for each cookie in the current request
- A PHP session is easily started by making a call to the session_start()function.This function first checks if a session is already started and if none is started then it starts one. It is recommended to put the call to session_start() at the beginning of the page.
- Session variables are stored in associative array called $_SESSION[]. These variables can be accessed during lifetime of a session.

8 a) Describe briefly the MVC architecture and the ORM used by rails. (8 M)

- Rails is Ruby based
- “A development framework for Web-based applications”
- Rails uses the Model-View-Controller architecture

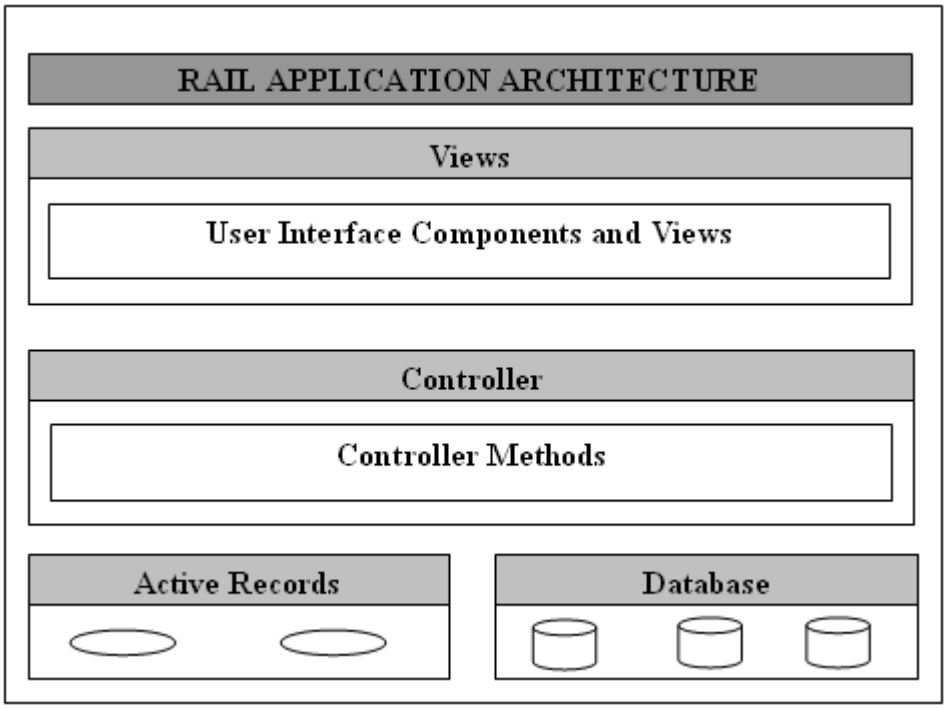
Model classes are the data classes, including constraint enforcement . It maintains the relationship between the objects and the database and handles validation, association, transactions, and more.

This subsystem is implemented in ActiveRecord library, which provides an interface and binding between the tables in a relational database and the Ruby program code that manipulates database records. Ruby method names are automatically generated from the field names of database tables.

View classes present the data to the user It is a presentation of data in a particular format, triggered by a controller's decision to present the data. They are script-based template systems like JSP, ASP, PHP, and very easy to integrate with AJAX technology.

This subsystem is implemented in ActionView library, which is an Embedded Ruby (ERb) based system for defining presentation templates for data presentation. Every Web connection to a Rails application results in the displaying of a view.

Controller classes perform computations and deal with user interaction: The facility within the application that directs traffic, on the one hand, querying the models for specific data, and on the other hand, organizing that data (searching, sorting, messaging it) into a form that fits the needs of a given view. This subsystem is implemented in ActionController, which is a data broker sitting between ActiveRecord (the database interface) and ActionView (the presentation engine).



- Rails use an Object-Relational Mapping approach to working with databases. ORM framework is written in an object oriented language and wrapped around a relational database. The object classes are mapped to the data tables in the database and the object instances are mapped to rows in those tables.

b) Write a note on methods in Ruby. (6 M)

- Methods can be defined outside classes
- When a method is called without an object reference, the default is self

Method syntax

def *name* [(*formal-parameters*)]

statements

end

- Parentheses can be omitted if there are no formal parameters

Return statement ends execution of the method

- With a value, it returns the value as the value of the method
- If no return ends a method, the last expression is the value of the method.

c) Explain the layouts with respect to rails.(6M)

- A layout template provides a standard template for other templates
  - Put in the layouts directory
  - Put a layout command in the ApplicationController class
- The layout template can provide header and styling directions while other templates provide content
- The @content_for_layout variable in the layout template provides the content from the other template
- A layout defines the surroundings of an HTML page. It's the place to define a common look and feel of your final output. Layout files reside in app/views/layouts.
- The process involves defining a layout template and then letting the controller know that it exists and to use it