USN | | | | | | | | | | |

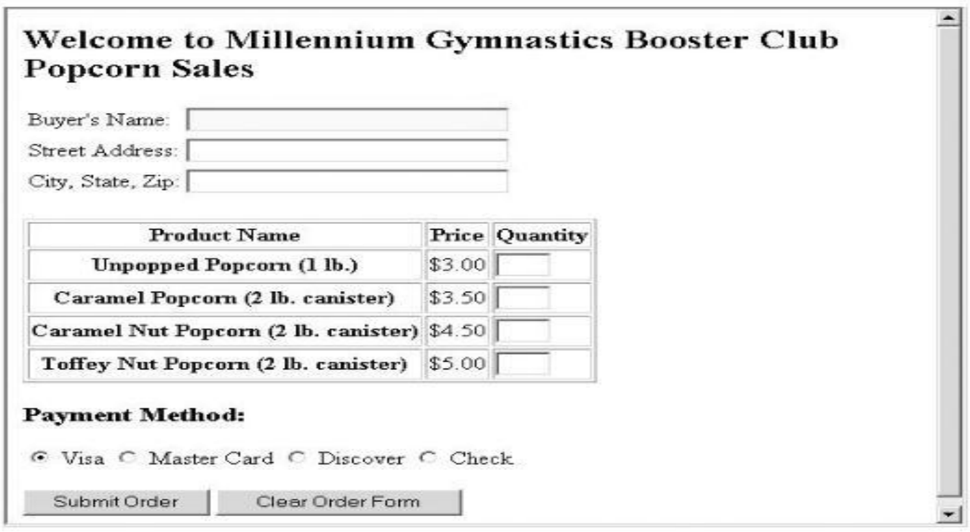
CMRIT — CMR INSTITUTE OF TECHNOLOGY, BENGALURU. ACCREDITED WITH A+ GRADE BY NAAC — CELEBRATING 25 YEARS

Internal Assessment Test 1 – Nov. 2017

Sub:	Web Technologies					Sub Code:	16MCA13	Branch:	MCA
Date:	8.11.2017	Duration:	90 min's	Max Marks:	50	Sem	I		OBE

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

		MARKS	CO	RBT
	PART I			
1 (a)	Explain request phase of HTTP?	[05]	CO1	L1
(b)	Explain Internet Protocol with appropriate example?	[05]	CO1	L1
	OR			
2 (a)	Explain response phase of HTTP?	[05]	CO1	L1
(b)	Explain DNS with neat diagram	[05]	CO1	L1
	PART II			
3 (a)	Describe syntactic difference between HTML and XHTML	[04]	CO2	L2
(b)	Explain the following tags with example i)Image ii)Link iii)List	[06]	CO2	L2
	OR			
4 (a)	Describe standard XHTML document structure with an example	[05]	CO2	L2
(b)	Create a web page using XHTML containing two vertical frames, with the right frame three times the width of the left frame. The right frame is further divided horizontally with top frame twice that of the bottom frame.	[05]	CO2	L2
	PART III			
5	Create a XHTML document that describes the form for taking orders for popcorn. Text boxes are used at the top of the form to collect the buyer's name and address. These are placed in a borderless table to force the text box align vertically. A second table to collect actual order. Each row of this table names a product, displays the price, and uses text box with size 2 to collect the quantity ordered using <td> tag. The payment method is input by the user through one of four radio buttons. Provide provision for submission of order and clear the order form.	[10]	CO2	L3



Welcome to Millennium Gymnastics Booster Club Popcorn Sales

Buyer's Name:	
Street Address:	
City, State, Zip:	

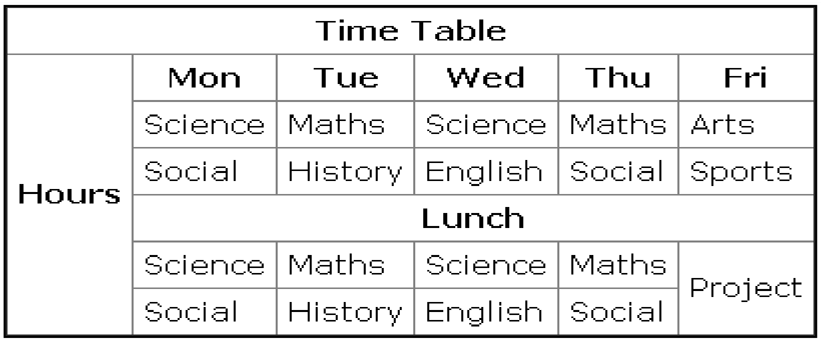
Product Name	Price	Quantity
Unpopped Popcorn (1 lb.)	$3.00	
Caramel Popcorn (2 lb. canister)	$3.50	
Caramel Nut Popcorn (2 lb. canister)	$4.50	
Toffey Nut Popcorn (2 lb. canister)	$5.00	

Payment Method:

[✓ Visa] Master Card Discover Check

OR

6	Write a XHTML program to create following table	[10]	CO1	L3



Time Table					
Hours	Mon	Tue	Wed	Thu	Fri
	Science	Maths	Science	Maths	Arts
	Social	History	English	Social	Sports
	Lunch				
	Science	Maths	Science	Maths	Project
	Social	History	English	Social	

PART IV

7	Explain the various levels of stylesheet with example for each.	[10]	CO2	L2
	OR			
8	Explain the various selector forms with examples.	[10]	CO1	L2

PART V

9	Write a program to illustrate all the CSS font and list properties.	[10]	CO2	L3
	OR			
10	Explain box model in CSS. Also Explain background image positioning in CSS	[10]	CO2	L3

Internal Assessment Test 1 – Nov. 2017

Sub:	Web Technologies					Sub Code:	16MCA13	Branch:	MCA
Date:	8.11.2017	Duration:	90 min's	Max Marks:	50	Sem	I		

PART I

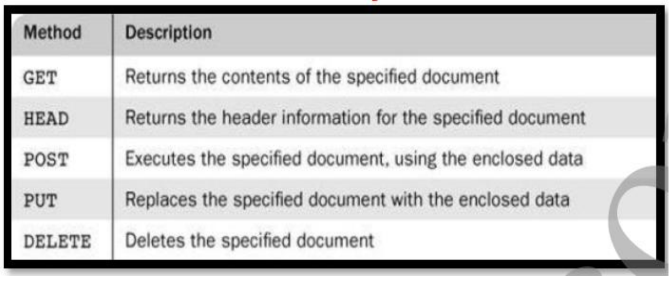
1 a) Explain request phase of HTTP? [05]

The general form of an HTTP request is as follows:
1. HTTP method Domain part of the URL HTTP version
2. Header fields
3. Blank line
4. Message body

The following is an example of the first line of an HTTP request:

GET /storefront.html HTTP/1.1

Table 1.1 HTTP request methods



Method	Description
GET	Returns the contents of the specified document
HEAD	Returns the header information for the specified document
POST	Executes the specified document, using the enclosed data
PUT	Replaces the specified document with the enclosed data
DELETE	Deletes the specified document

The format of a header field is the field name followed by a colon and the value of the field. There are four categories of header fields:
1. General: For general information, such as the date
2. Request: Included in request headers
3. Response: For response headers
4. Entity: Used in both request and response headers

Accept: text/plain
Accept: text/html
Can be written as
Accept: text/*

A wildcard character, the asterisk (*), can be used to specify that part of a MIME type can be anything.
The Host: host name request field gives the name of the host. The Host field is required for HTTP1.1.
The If-Modified-Since: date request field specifies that the requested file should be sent only if it has been modified since the given date. If the request has a body, the length of that body must be given with a Content-length field. The header of a request must be followed by a blank line, which is used to separate the header from the body of the request.

b) Explain Internet Protocol with appropriate example?[05]

Internet Protocol Addresses

The Internet Protocol (IP) address of a machine connected to the Internet is a unique 32-bit number.

- IP addresses usually are written (and thought of) as four 8-bit numbers, separated by periods.
- The four parts are separately used by Internet-routing computers to decide where a message must go next to get to its destination.
- Although people nearly always type domain names into their browsers, the IP works just as well.
- For example, the IP for United Airlines (`www.ual.com`) is `209.87.113.93`. So, if a browser is pointed at `http://209.87.113.93`, it will be connected to the United Airlines Web site.

OR

2 a) Explain response phase of HTTP? [05]

The general form of an HTTP response is as follows:

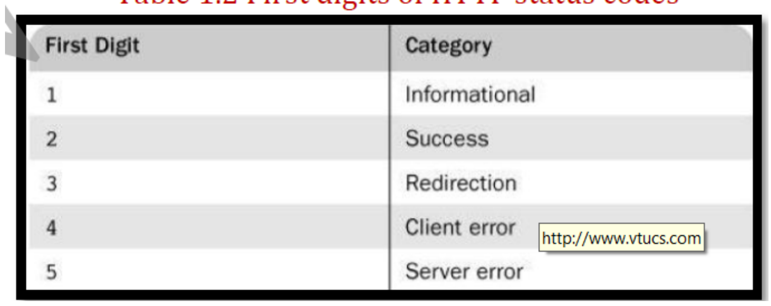
1. Status line
2. Response header fields
3. Blank line
4. Response body

The status line includes the HTTP version used, a three-digit status code for the response, and a short textual explanation of the status code. For example, most responses begin with the following:

HTTP/1.1 200 OK

The status codes begin with 1, 2, 3, 4, or 5. The general meanings of the five categories specified by these first digits are shown in Table 1.2.

Table 1.2 First digits of HTTP status codes



First Digit	Category
1	Informational
2	Success
3	Redirection
4	Client error
5	Server error

One of the more common status codes is one user never want to see: 404 Not Found, which means the requested file could not be found.

b) Explain DNS with neat diagram [05]

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** .

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:

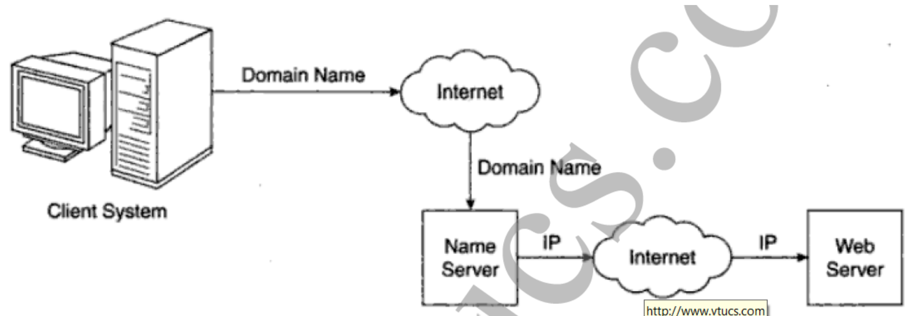
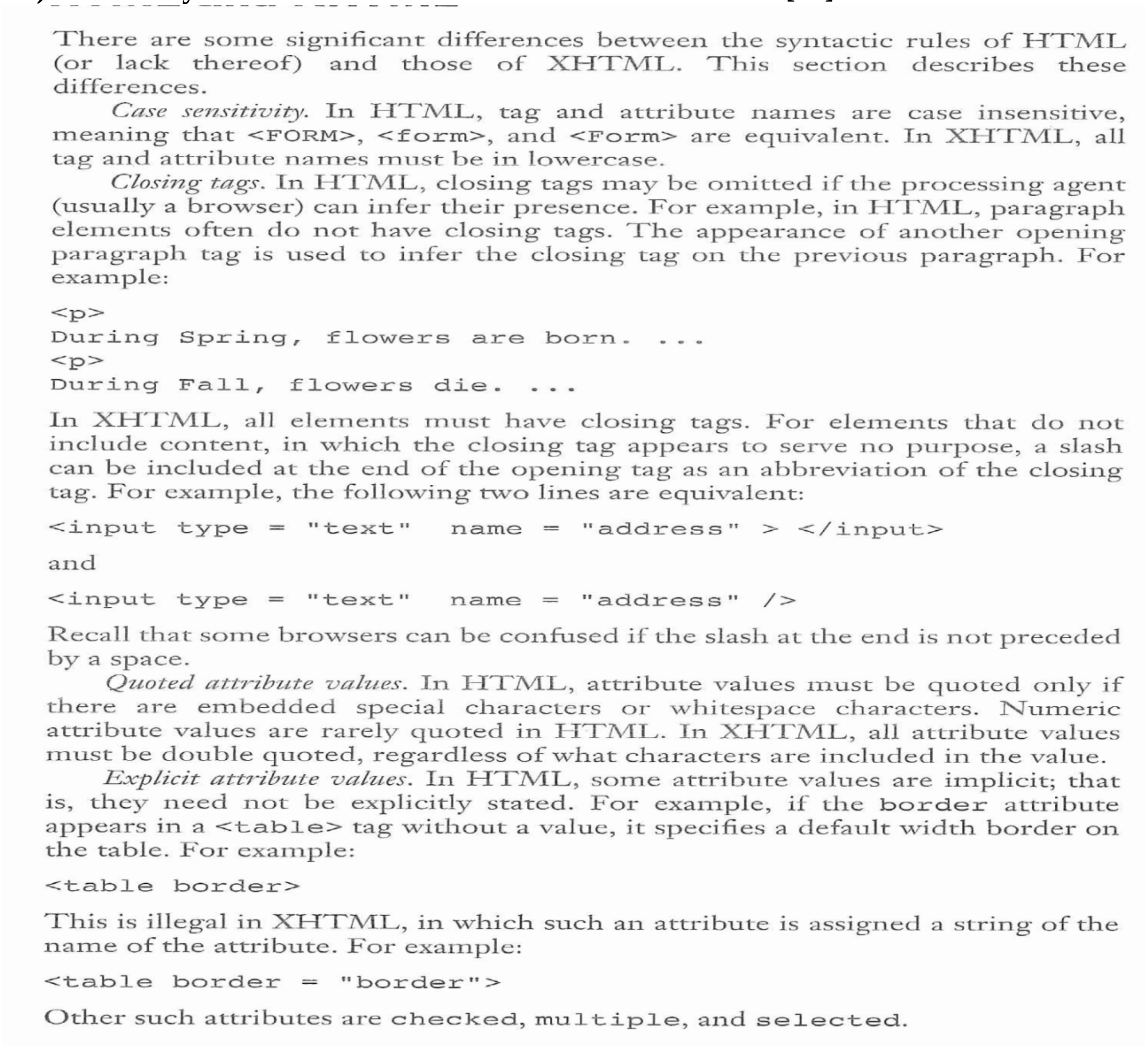


Figure 1.1 Domain name conversion

PART II

3 a) Describe syntactic difference between HTML and XHTM [04]


There are some significant differences between the syntactic rules of HTML (or lack thereof) and those of XHTML. This section describes these differences.

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 tags may be omitted if the processing agent (usually a browser) can infer their presence. For example, in HTML, paragraph elements often do not have closing tags. The appearance of another opening paragraph tag is used to infer the closing tag on the previous paragraph. For example:

```
<p>
During Spring, flowers are born. ...
<p>
During Fall, flowers die. ...
```

In XHTML, all elements must have closing tags. For elements that do not include content, in which the closing tag appears to serve no purpose, a slash can be included at the end of the opening tag as an abbreviation of the closing tag. For example, the following two lines are equivalent:

```
<input type = "text"  name = "address" > </input>
```

and

```
<input type = "text"  name = "address" />
```

Recall that some browsers can be confused if the slash at the end is not preceded by a space.

Quoted attribute values. In HTML, attribute values must be quoted only if there are embedded special characters or whitespace characters. Numeric attribute values are rarely quoted in HTML. In XHTML, all attribute values must be double quoted, regardless of what characters are included in the value.

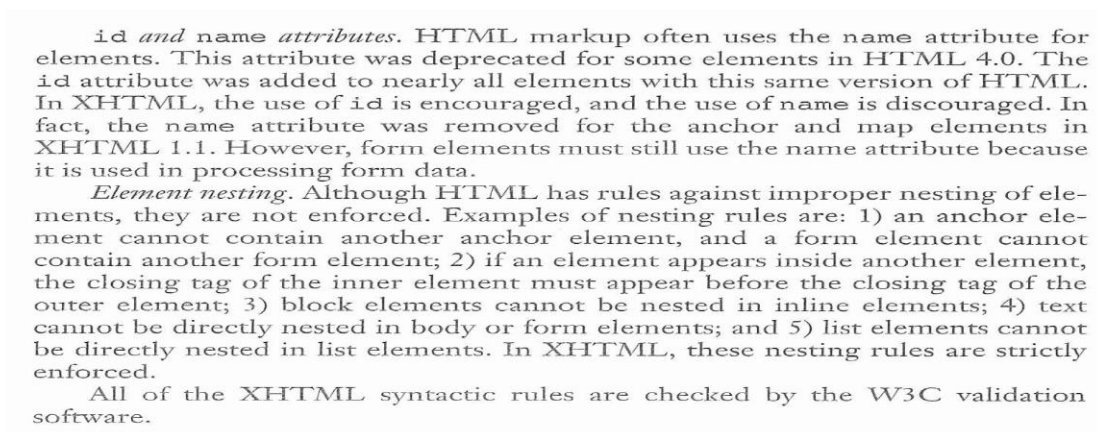
Explicit attribute values. In HTML, some attribute values are implicit; that is, they need not be explicitly stated. For example, if the `border` attribute appears in a `<table>` tag without a value, it specifies a default width border on the table. For example:

```
<table border>
```

This is illegal in XHTML, in which such an attribute is assigned a string of the name of the attribute. For example:

```
<table border = "border">
```

Other such attributes are `checked`, `multiple`, and `selected`.



`id` *and* `name` *attributes*. HTML markup often uses the `name` attribute for elements. This attribute was deprecated for some elements in HTML 4.0. The `id` attribute was added to nearly all elements with this same version of HTML. In XHTML, the use of `id` is encouraged, and the use of `name` is discouraged. In fact, the `name` attribute was removed for the anchor and map elements in XHTML 1.1. However, form elements must still use the `name` attribute because it is used in processing form data.

Element nesting. Although HTML has rules against improper nesting of elements, they are not enforced. Examples of nesting rules are: 1) an anchor element cannot contain another anchor element, and a form element cannot contain another form element; 2) if an element appears inside another element, the closing tag of the inner element must appear before the closing tag of the outer element; 3) block elements cannot be nested in inline elements; 4) text cannot be directly nested in body or form elements; and 5) list elements cannot be directly nested in list elements. In XHTML, these nesting rules are strictly enforced.

All of the XHTML syntactic rules are checked by the W3C validation software.

b) Explain the following tags with example [05]
i)Image ii)Link iii)List

i)Image

- Image can be displayed on the web page using <img> tag.
- When the <img> tag is used, it should also be mentioned which image needs to be displayed. This is done using src attribute.
- Attribute means extra information given to the browser
- Whenever <img> tag is used, alt attribute is also used.
- Alt stands for alert.
- Some very old browsers would not be having the capacity to display the images.
- In this case, whatever is the message given to alt attribute, that would be displayed.
- Another use of alt is □ when image display option has been disabled by user. The option is normally disabled when the size of the image is huge and takes time for downloading.

```
<html>
        <head> <title>display image</title> </head>
        <body>
                <img src="java.png" alt="cannot display"/>
         </body>
</html>
```

ii) Link

Hyperlinks are the mechanism which allows the navigation from one page to another.
The term “hyper” means beyond and “link” means connection
Whichever text helps in navigation is called hypertext
Hyperlinks cam be created using <a> (anchor tag)
The attribute that should be used for <a> is href
Program: hyper.html

```
<html>
        <head>
                <title> hyperlink </title>
        </head>
        <a href = "link.html"> CLICK HERE </a>
</html>
```

iii)List

<u>Unordered Lists:</u>

The <ul> tag, which is a block tag, creates an unordered list. Each item in a list is specified with an <li> tag (li is an acronym for list item). Any tags can appear in a list item, including nested lists. When displayed, each list item is implicitly preceded by a bullet.

```
<html>
        <head> <title> Unordered     List </title> </head>
        <body>
                <h1Engine Aircraft </h1>
                 <ul>
                        <li>Cessna Skyhawk</li>
                        <li>Beechcraft Bonanza</li>
                </ul>
        </body>
</html>
```

Ordered Lists:

Ordered lists are lists in which the order of items is important. This orderedness of a list is shown in the display of the list by the implicit attachment of a sequential value to the beginning of each item. The default sequential values are Arabic numerals, beginning with 1. An ordered list is created within the block tag <ol>. The items are specified and displayed just as are those in unordered lists, except that the items in an ordered list are preceded by sequential values instead of bullets.

```
<html>
        <head> <title> Unordered     List </title> </head>
        <body>
                <h1Engine Aircraft </h1>
                 <ol>
                        <li>Cessna Skyhawk</li>
                        <li>Beechcraft Bonanza</li>
                </ol>
        </body>
</html>
```

Definition Lists:

As the name implies, definition lists are used to specify lists of terms and their definitions, as in glossaries. A definition list is given as the content of a <dl> tag, which is a block tag. Each term to be defined in the definition list is given as the content of a <dt> tag. The definitions themselves are specified as the content of <dd> tags. The defined terms of a definition list are usually displayed in the left margin; the definitions are usually shown indented on the line or lines following the term.

```
<html>
        <head> <title> Definition List </title> </head>
        <body>
                 <h1> Cessna Skyhawk </h1>
                <dl>
                         <dt>152 </dt>
                        <dd>Two-place trainer</dd>
                        <dt> 172 </dt>
                        <dd>Smaller four-place airplane</dd>
                         <dt> 120 </dt>
                         <dd>Six-place airplane- high performance</dd>
                </dl>
        </body>
</html>
```

OR

4 a) Describe standard XHTML document structure with an example [05]

Standard XHTML Document Structure

- Every XHTML document must begin with an xml declaration element that simply identifies the document as being one based on XML. This element includes an attribute that specifies the version number 1.0.
- The xml declaration usually includes a second attribute, encoding, which specifies the encoding used for the document [utf-8].
- Following is the xml declaration element, which should be the first line of every XHTML document:

  <?xml version = "1.0" encoding = "utf-8"?>
- Note that this declaration must begin in the first character position of the document file.
- The xml declaration element is followed immediately by an SGML DOCTYPE command, which specifies the particular SGML document-type definition (DTD) with which the document complies, among other things.
- The following command states that the document in which it is included complies with the XHTML 1.0 Strict standard:

  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml1-strict.dtd">
- An XHTML document must include the four tags <html>, <head>, <title>, and <body>.
- The <html> tag identifies the root element of the document. So, XHTML documents always have an <html> tag immediately following the DOCTYPE command, and they always end with the closing html tag, </html>.
- The html element includes an attribute, xmlns, that specifies the XHTML namespace, as shown in the following element:

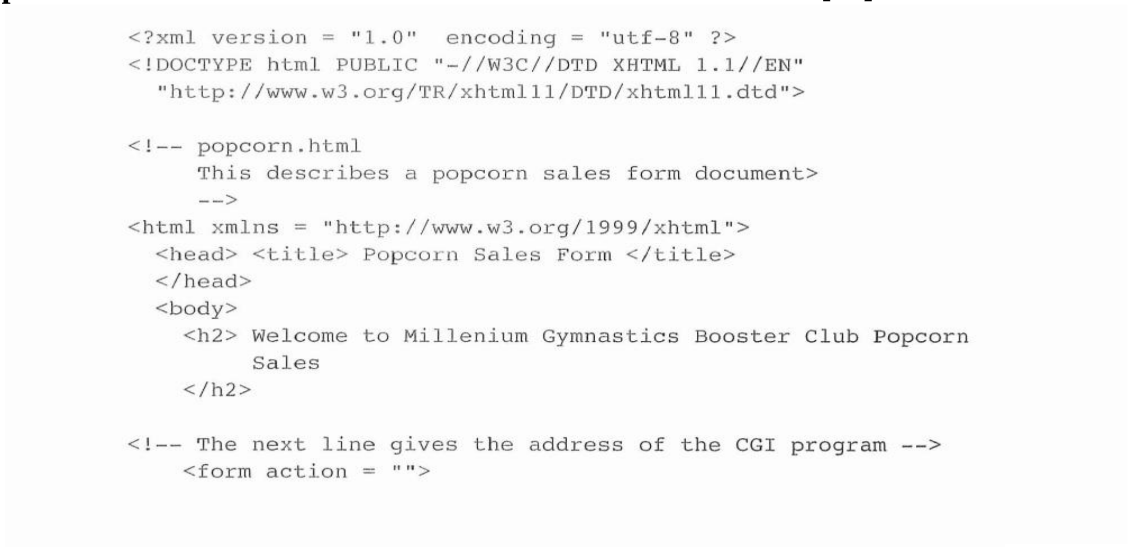
  <html xmlns = "http://www.w3.org/1999/xhtml">
- Although the xmlns attribute's value looks like a URL, it does not specify a document. It is just a name that happens to have the form of a URL.
- An XHTML document consists of two parts, named the head and the body.
- The <head> element contains the head part of the document, which provides information about the document and does not provide the content of the document.
- The body of a document provides the content of the document.
- The content of the title element is displayed by the browser at the top of its display window, usually in the browser window's title bar.

b) Create a web page using XHTML containing two vertical frames, with the right frame three times the width of the left frame. The right frame is further divided horizontally with top frame twice that of the bottom frame. [05]

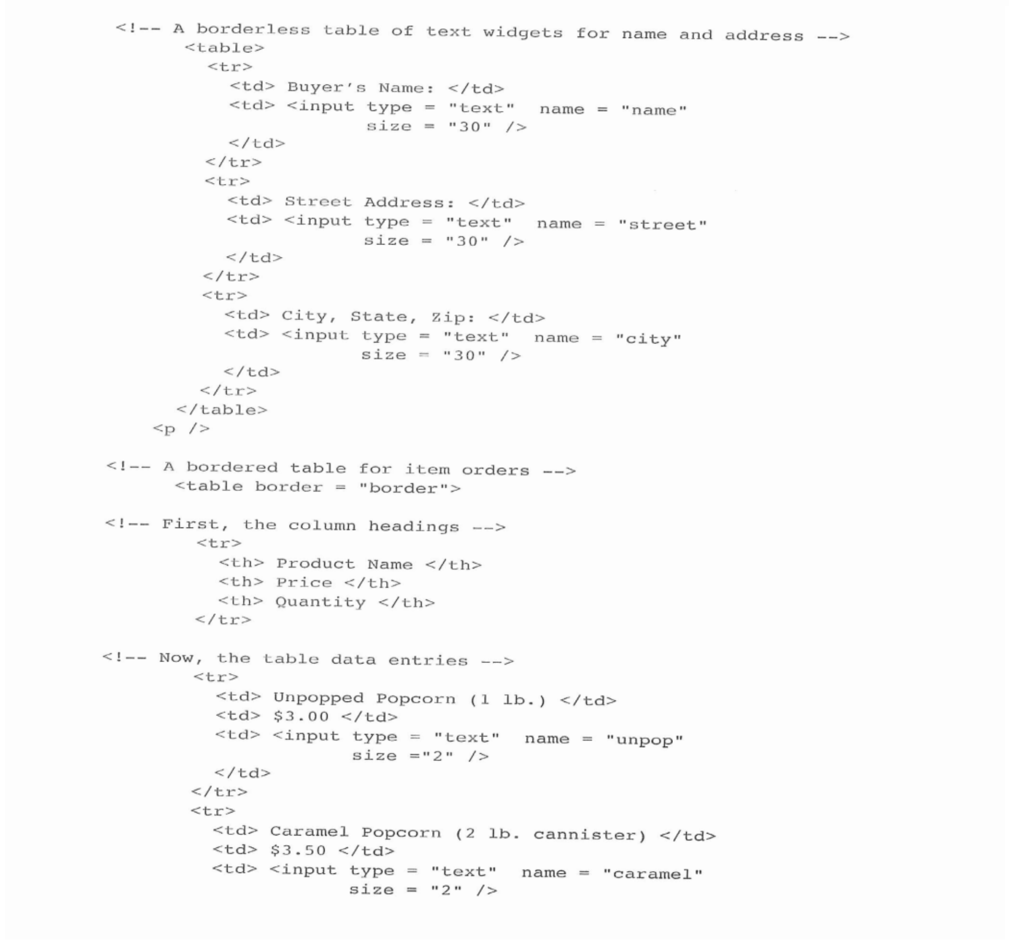
```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
        <head>
                <title>Time table</title>
        </head>
        <frameset cols="*,3*">
                <frame src="file1.html" />
                <frameset rows="2*,*">
                        <frame src="file2.html" />
                        <frame src="file3.html" />
                </frameset>
        </frameset>

</html>
```

PART III

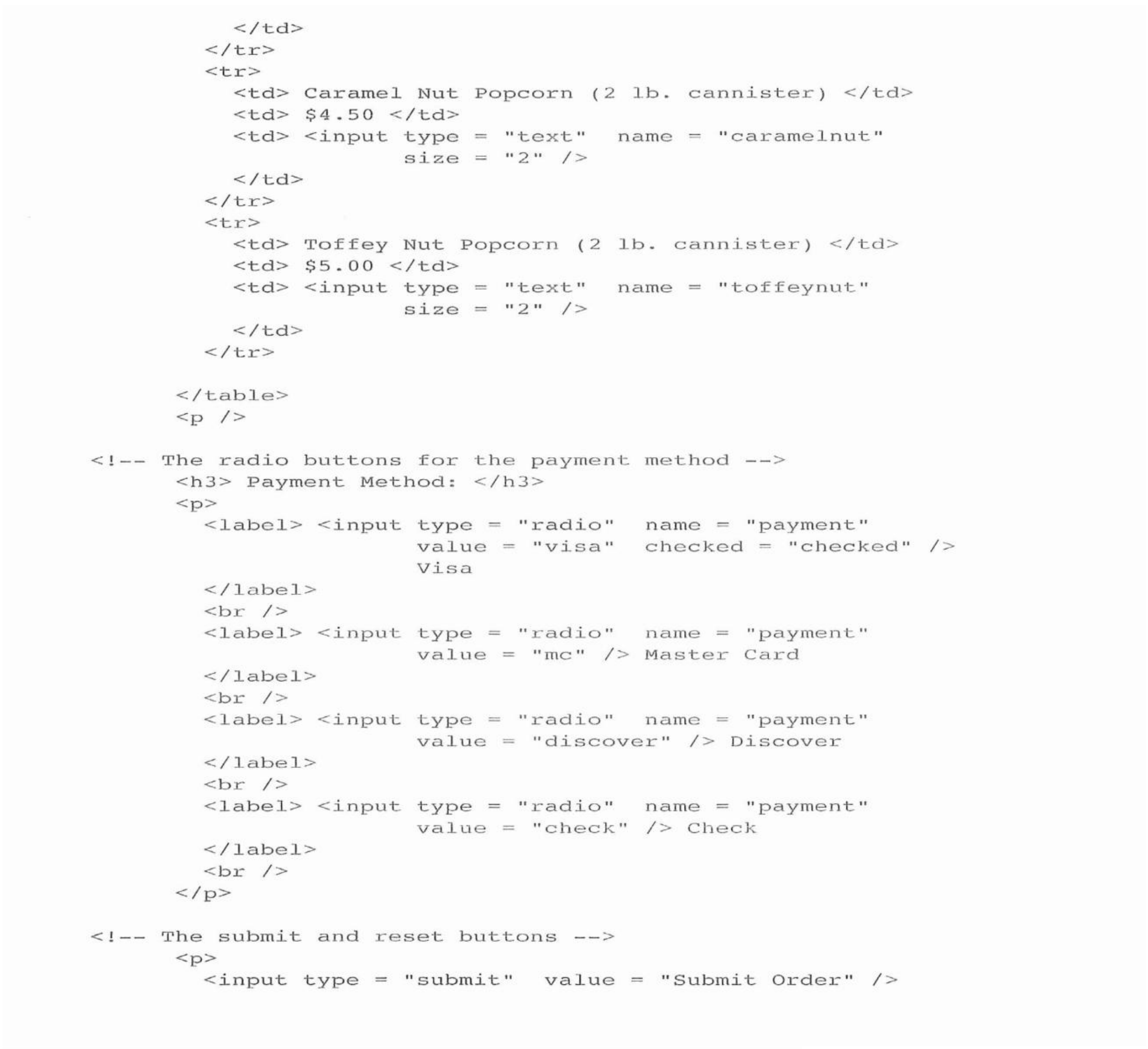
5) Create a XHTML document that describes the form for taking orders for popcorn. Text boxes are used at the top of the form to collect the buyer’s name and address. These are placed in a borderless table to force the text box align vertically. A second table to collect actual order. Each row of this table names a product, displays the price, and uses text box with size 2 to collect the quantity ordered using <td> tag. The payment method is input by the user through one of four radio buttons. Provide provision for submission of order and clear the order form. [10]



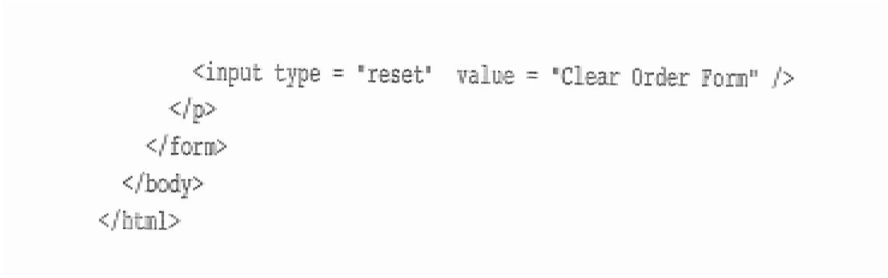
```
<?xml version = "1.0"  encoding = "utf-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!-- popcorn.html
     This describes a popcorn sales form document>
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Popcorn Sales Form </title>
  </head>
  <body>
    <h2> Welcome to Millenium Gymnastics Booster Club Popcorn
         Sales
    </h2>

<!-- The next line gives the address of the CGI program -->
    <form action = "">
```



```
<!-- A borderless table of text widgets for name and address -->
      <table>
        <tr>
          <td> Buyer's Name: </td>
          <td> <input type = "text"  name = "name"
                      size = "30" />
          </td>
        </tr>
        <tr>
          <td> Street Address: </td>
          <td> <input type = "text"  name = "street"
                      size = "30" />
          </td>
        </tr>
        <tr>
          <td> City, State, Zip: </td>
          <td> <input type = "text"  name = "city"
                      size = "30" />
          </td>
        </tr>
      </table>
    <p />

<!-- A bordered table for item orders -->
      <table border = "border">

<!-- First, the column headings -->
        <tr>
          <th> Product Name </th>
          <th> Price </th>
          <th> Quantity </th>
        </tr>

<!-- Now, the table data entries -->
        <tr>
          <td> Unpopped Popcorn (1 lb.) </td>
          <td> $3.00 </td>
          <td> <input type = "text"  name = "unpop"
                      size ="2" />
          </td>
        </tr>
        <tr>
          <td> Caramel Popcorn (2 lb. cannister) </td>
          <td> $3.50 </td>
          <td> <input type = "text"  name = "caramel"
                      size = "2" />
```



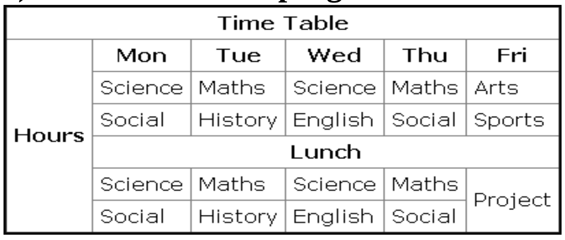
```
            </td>
          </tr>
          <tr>
            <td> Caramel Nut Popcorn (2 lb. cannister) </td>
            <td> $4.50 </td>
            <td> <input type = "text"  name = "caramelnut"
                        size = "2" />
            </td>
          </tr>
          <tr>
            <td> Toffey Nut Popcorn (2 lb. cannister) </td>
            <td> $5.00 </td>
            <td> <input type = "text"  name = "toffeynut"
                        size = "2" />
            </td>
          </tr>

        </table>
        <p />

<!-- The radio buttons for the payment method -->
        <h3> Payment Method: </h3>
        <p>
          <label> <input type = "radio"  name = "payment"
                         value = "visa"  checked = "checked" />
                         Visa
          </label>
          <br />
          <label> <input type = "radio"  name = "payment"
                         value = "mc" /> Master Card
          </label>
          <br />
          <label> <input type = "radio"  name = "payment"
                         value = "discover" /> Discover
          </label>
          <br />
          <label> <input type = "radio"  name = "payment"
                         value = "check" /> Check
          </label>
          <br />
        </p>

<!-- The submit and reset buttons -->
        <p>
          <input type = "submit"  value = "Submit Order" />
```



```
          <input type = "reset"  value = "Clear Order Form" />
        </p>
      </form>
    </body>
  </html>
```

OR

6) Write a XHTML program to create following table[10]



Time Table					
Hours	Mon	Tue	Wed	Thu	Fri
	Science	Maths	Science	Maths	Arts
	Social	History	English	Social	Sports
	Lunch				
	Science	Maths	Science	Maths	Project
	Social	History	English	Social	

```
<?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">

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

```
    <head>
        <title>Time table</title>
    </head>
    <body>

        <table border="1" cellpadding="0" cellspacing="0">
            <tr align="center">
                <th colspan="6">Time Table </th>
            </tr>
            <tr>
              <th rowspan="6">Hours</th>
              <th>Mon</th>
              <th>Tue</th>
              <th>Wed</th>
              <th>Thu</th>
              <th>Fri</th>
            </tr>
            <tr>
              <td>Science</td>
              <td>Maths</td>
              <td>Science</td>
              <td>Maths</td>
              <td>Arts</td>
            </tr>
            <tr>
              <td>Social</td>
              <td>History</td>
              <td>English</td>
              <td>Social</td>
              <td>Sports</td>
            </tr>
            <tr align="center">
                <td colspan="5">Lunch</td>
            </tr>
            <tr>
              <td>Science</td>
              <td>Maths</td>
              <td>Science</td>
              <td>Maths</td>
              <td rowspan="2">Project</td>
            </tr>
            <tr>
              <td>Social</td>
              <td>History</td>
              <td>English</td>
              <td>Social</td>
            </tr>
        </table>
    </body>
</html>
```

PART IV

7) Explain the various levels of stylesheet with example for each. [10]

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>
        <body>
                <h1>External stylesheet</h1>
        </body>
</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>Document level stylesheet </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;"> Inline stylesheet</h1>
</html>
```

OR

8) Explain the various selector forms with examples. [10]

1. <u>Simple Selector Forms:</u>

In case of simple selector, a tag is used. If the properties of the tag are changed, then it reflects at all the places when used in the program. The selector can be any tag. If the new properties for a tag are not mentioned within the rule list, then the browser uses default behavior of a tag.

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

2. Class Selectors:
   In class selector, it is possible to give different properties for different elements

```
 <style type = "text/css">
           p.one {
                       font-family: 'Lucida Handwriting';
                       font-size: 25pt;
                       color: Red;
                      }
</style>
```

3. Generic Selectors:
   In case of generic selector, when the class is created, it would not be associated to any particular tag. In other words, it is generic in nature.

```
<style type = "text/css">
            .one {
                      font-family: 'Monotype Corsiva';
                      color: green;
                  }
</style>
```

4. id Selectors:
   An id selector allows the application of a style to one specific element.

```
<style type = "text/css">
            #one {
                      font-family: 'lucida calligraphy';
                      color: purple;
                      }
</style>
```

5. Universal Selectors:
   The universal selector, denoted by an asterisk (*), applies its style to all elements in a document.

```
<style type = "text/css">
           * {
                      font-family: 'lucida calligraphy';
                      color: purple;
                      }
 </style>
```

6. Pseudo Classes:
   Pseudo class selectors are used if the properties are to be changed dynamically. For example: when mouse movement happens, in other words, hover happens or focus happens.

```
           <style type = "text/css">
                       input:focus
                      {
```

```
            font-family: 'lucida calligraphy';
            color: purple;
            font-size:100;
        }
         input:hover {
             font-family: 'lucida handwriting';
            color: violet;
            font-size:40;
        }
```

PART V

9) Write a program to illustrate all the CSS font and list properties. [10]

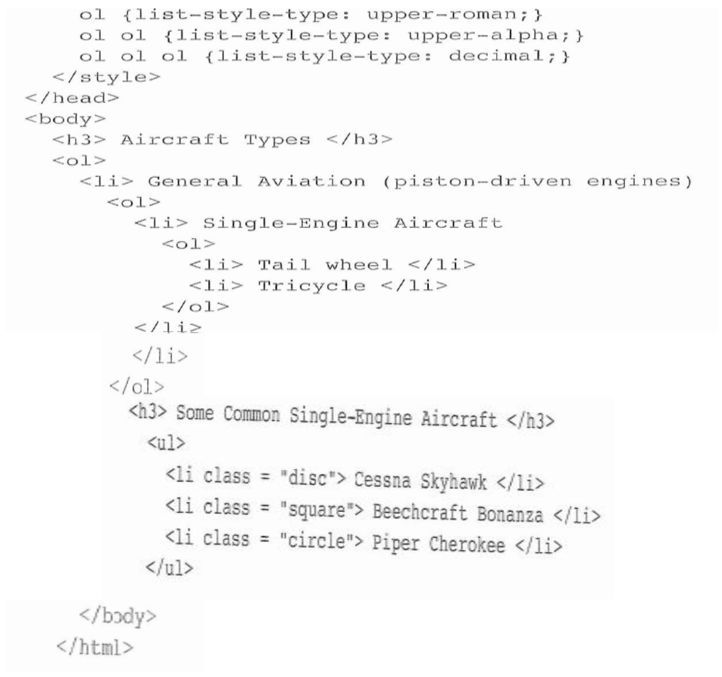
Font Properties

```
<?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">

<!-- fonts.html
     An example to illustrate font properties
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Font properties </title>
    <style type = "text/css">
      p.major {font-size: 14pt;
               font-style: italic;
               font-family: 'Times New Roman';
              }
      p.minor {font: 10pt bold 'Courier New';}
      h2 {font-family: 'Times New Roman';
          font-size: 24pt; font-weight: bold}
      h3 {font-family: 'Courier New'; font-size: 18pt}
    </style>
  </head>
  <body>
    <p class = "major">
      If a job is worth doing, it's worth doing right.
    </p>
    <p class = "minor">
      Two wrongs don't make a right, but they certainly
      can get you in a lot of trouble.
    </p>
    <h2> Chapter 1 Introduction </h2>
    <h3> 1.1 The Basics of Computer Networks </h3>
  </body>
</html>
```

List properties

```
<?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">

<!-- sequence_types.html
     An example to illustrate sequence type styles
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Sequence types </title>
```

```
    <style type = "text/css">
      li.disc {list-style-type: disc}
      li.square {list-style-type: square}
```



```
      ol {list-style-type: upper-roman;}
      ol ol {list-style-type: upper-alpha;}
      ol ol ol {list-style-type: decimal;}
    </style>
  </head>
  <body>
    <h3> Aircraft Types </h3>
    <ol>
      <li> General Aviation (piston-driven engines)
        <ol>
          <li> Single-Engine Aircraft
            <ol>
              <li> Tail wheel </li>
              <li> Tricycle </li>
            </ol>
          </li>
          </li>
        </ol>
          <h3> Some Common Single-Engine Aircraft </h3>
          <ul>
            <li class = "disc"> Cessna Skyhawk </li>
            <li class = "square"> Beechcraft Bonanza </li>
            <li class = "circle"> Piper Cherokee </li>
          </ul>

      </body>
    </html>
```

OR

10) Explain box model in CSS. Also Explain background image positioning in CSS[10]

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

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.

BACKGROUND IMAGES

The background-image property is used to place an image in the background of an element.

The background image can be replicated as necessary to fill the area of the element. This replication is called tiling. Tiling can be controlled with the background-repeat property, which can take the value repeat (the default), no-repeat, repeat-x, or repeat-y. The no-repeat value specifies that just one copy of the image is to be displayed. The repeat-x value means that the image is to be repeated horizontally; repeat-y means that the image is to be repeated vertically. In addition, the position of a non-repeated background image can be specified with the background-position property, which can take a large number of different values. The keyword values are top, center, bottom, left, and right, all of which can be used in many different combinations.