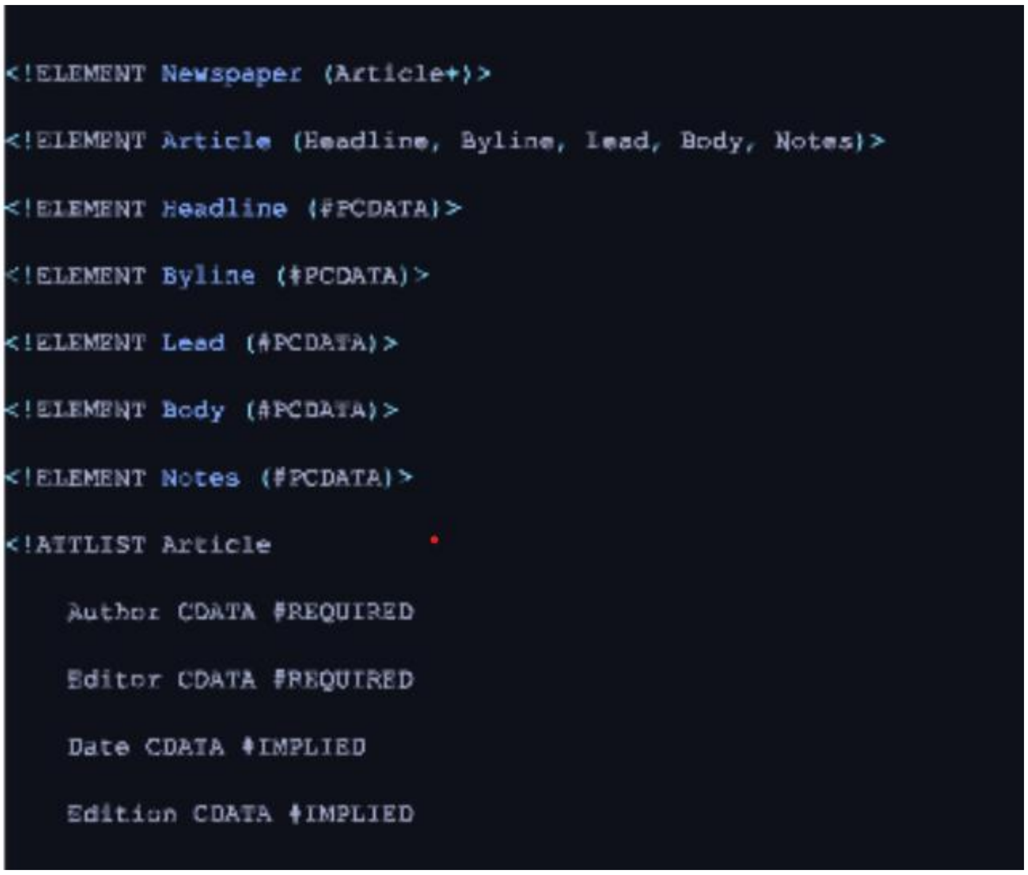
Assignment: 4

Q-1: Create an external DTD file for a Newspaper information satisfying the following requirements (elements and attribute names are given in italics, Newspaper is the root):

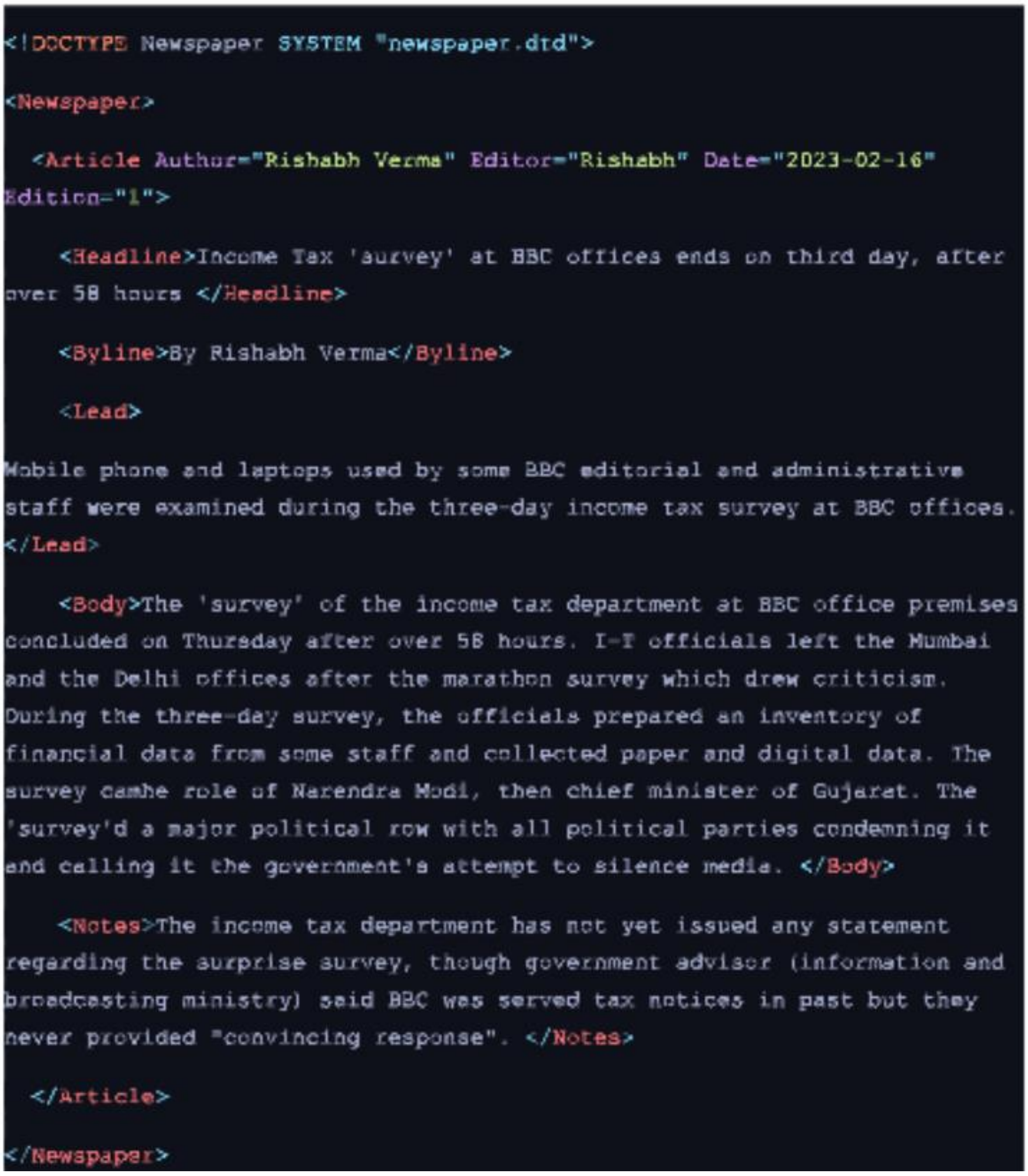
• A Newspaper must contain at least one Article. • Each Article should be a sequence (one and exactly one sequence) of Headline, Byline, Lead, Body, and Notes. • Each Headline, Byline, Lead, Body, and Notes will contain parseable character data. • The Article will have four attributes, Author, Editor, Date, and Edition. Among them, Author and Editor are strictly required, and, Date and Edition are optional.

Write an example XML file satisfying the necessary constraints, and validate the file against the DTD file



```
<!ELEMENT Newspaper (Article+)>
<!ELEMENT Article (Headline, Byline, Lead, Body, Notes)>
<!ELEMENT Headline (#PCDATA)>
<!ELEMENT Byline (#PCDATA)>
<!ELEMENT Lead (#PCDATA)>
<!ELEMENT Body (#PCDATA)>
<!ELEMENT Notes (#PCDATA)>
<!ATTLIST Article
    Author CDATA #REQUIRED
    Editor CDATA #REQUIRED
    Date CDATA #IMPLIED
    Edition CDATA #IMPLIED
```

Here's a XML file that satisfies the constraints:



```xml
<!DOCTYPE Newspaper SYSTEM "newspaper.dtd">
<Newspaper>
  <Article Author="Rishabh Verma" Editor="Rishabh" Date="2023-02-16"
Edition="1">
    <Headline>Income Tax 'survey' at BBC offices ends on third day, after
over 58 hours </Headline>
    <Byline>By Rishabh Verma</Byline>
    <Lead>
Mobile phone and laptops used by some BBC editorial and administrative
staff were examined during the three-day income tax survey at BBC offices.
</Lead>
    <Body>The 'survey' of the income tax department at BBC office premises
concluded on Thursday after over 58 hours. I-T officials left the Mumbai
and the Delhi offices after the marathon survey which drew criticism.
During the three-day survey, the officials prepared an inventory of
financial data from some staff and collected paper and digital data. The
survey camhe role of Narendra Modi, then chief minister of Gujarat. The
'survey'd a major political row with all political parties condemning it
and calling it the government's attempt to silence media. </Body>
    <Notes>The income tax department has not yet issued any statement
regarding the surprise survey, though government advisor (information and
broadcasting ministry) said BBC was served tax notices in past but they
never provided "convincing response". </Notes>
  </Article>
</Newspaper>
```

2. In the following example of XML file having CATALOG as root element. Create a DTD file that satisfying the xml document.

```xml
<?xml version="1.0"?>
<CATALOG>
<PRODUCT NAME="" CATEGORY="HandTool" PARTNUM="" PLANT="Chicago"
INVENTORY="InStock">
<SPECIFICATIONS WEIGHT="" POWER="">1600CC</SPECIFICATIONS>
<OPTIONS FINISH="Matte" ADAPTER="Included" CASE="HardShell">4</OPTIONS>
<PRICE MSRP="" WHOLESALE="" STREET="" SHIPPING="">100</PRICE>
<NOTES>Hello</NOTES>
</PRODUCT>
</CATALOG>
```

Answer:

```
<!DOCTYPE CATALOG [
  <!ELEMENT CATALOG (PRODUCT+)>
  <!ELEMENT PRODUCT (SPECIFICATIONS, OPTIONS, PRICE, NOTES)>
  <!ATTLIST PRODUCT
    NAME CDATA #IMPLIED
    CATEGORY CDATA #IMPLIED
    PARTNUM CDATA #IMPLIED
    PLANT CDATA #IMPLIED
    INVENTORY CDATA #IMPLIED
  >
  <!ELEMENT SPECIFICATIONS (#PCDATA)>
  <!ATTLIST SPECIFICATIONS
    WEIGHT CDATA #IMPLIED
    POWER CDATA #IMPLIED
  >
  <!ELEMENT OPTIONS (#PCDATA)>
  <!ATTLIST OPTIONS
    FINISH CDATA #IMPLIED
    ADAPTER CDATA #IMPLIED
    CASE CDATA #IMPLIED
  >
  <!ELEMENT PRICE (#PCDATA)>
  <!ATTLIST PRICE
    MSRP CDATA #IMPLIED
    WHOLESALE CDATA #IMPLIED
    STREET CDATA #IMPLIED
    SHIPPING CDATA #IMPLIED
  >
  <!ELEMENT NOTES (#PCDATA)>
]>
```

Assignment: 5

1. Create a W3C schema about information regarding the characters described in a book satisfying the following requirements: • book will be the root element. • book must have an attribute named isbn, which must be a number having 10 digits. • The root element will have title (must occur exactly once), author (must occur exactly once), character (may occur any number of times including 0) elements in sequence. • The title is a string. • The author is a string. • Each character will further have sub-elements name (must occur exactly once), type (must occur exactly once), and location (must occur exactly once) in sequence. • name is a string and can be of atmost length 10. This element can have an optional attribute king-of, whose value can be string. The value of king-of must start with a capital letter followed by lowercase letters or space, and of atleast length 3. • type is a string that can take values only from Men, Elves, Dwarves, or Orcs. • location must have the fixed value of Middle-earth.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="book">
    <xs:complexType>
    <xs:sequence>
    <xs:element name="title" type="xs:string"/>
    <xs:element name="author" type="xs:string"/>
    <xs:element name="character" minOccurs="0" maxOccurs="unbounded">
           <xs:complexType>
            <xs:sequence>
              <xs:element name="name" type="xs:string">
                <xs:complexType>
        <xs:attribute name="king-of" type="kingOfType" use="optional"/>
                </xs:complexType>
              </xs:element>
              <xs:element name="type" type="characterType"/>
    <xs:element name="location" type="xs:string" fixed="Middle-earth"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="isbn" type="xs:integer"/>
    </xs:complexType>
  </xs:element>

```

```

<xs:simpleType name="kingOfType">
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z][a-z ]{2,}"/>
    </xs:restriction>
 <xs:simpleType>

  <xs:simpleType name="characterType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Men"/>
      <xs:enumeration value="Elves"/>
      <xs:enumeration value="Dwarves"/>
      <xs:enumeration value="Orcs"/>
    </xs:restriction>
  </xs:simpleType>

</xs:schema>

```

2. Prepare an XML file conforming to the schema desribed above with atleast two characters, and prove the vaidity using xmllint application

```
<?xmlversion="1.0"encoding="UTF-8"?>
<bookisbn="1234567890">
    <title>TheLordoftheRings</title>
    <author>J.R.R.Tolkien</author>

    <character><nameking-of="TheShire">Frodo</name>
    <type>Hobbit</type>
    <location>Middle-earth</location>
    </character>
        <character>
        <nameking-of="Wizards">Gandalf</name>
        <type>Maia</type>
        <location>Middle-earth</location>
        </character>
</book>
To validate this XML file using xmllint, xmllint--schema  book.xsd book.xml
```

Assignment : 6

Write an xml dom to display the value of title element for cooking category of book element in books.xml document.

Book.xml

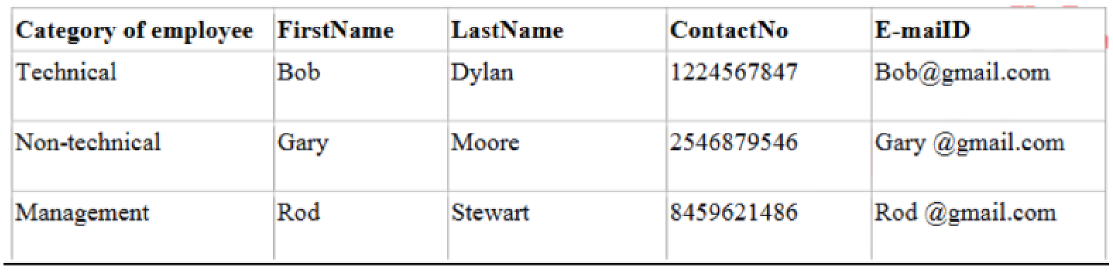
```xml
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
<book>
<category>cooking</category>
<title>Everyday Italian</title>
<author>Giada De Laurentiis</author>
<year>2005</year>
<price>30.00</price>
</book>
<book>
<category>Children</category>
<title>Harry Potter</title>
<author>J K. Rowling</author>
<year>2005</year>
<price>29.99</price>
</book>
</bookstore>
```

Book.html

```html
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>
<script>
    var xmlDoc,i;
if(window.XMLHttpRequest){
xmlDoc = new XMLHttpRequest();
}
else{
xmlDoc = new ActiveXObject("Microsoft.XMLHTTP")
}
xmlDoc.open("GET","book.xml",false);
xmlDoc.send();
var docxml = xmlDoc.responseXML;
var x = docxml.getElementsByTagName("book");
for(i=0; i < x.length;i++){
```

```
var cat = x[i].getElementsByTagName("category") [0].childNodes[0].nodeValue;
var title=x[i].getElementsByTagName("title")[0].childNodes[0].nodeValue;
var author=x[i].getElementsByTagName("author") [0].childNodes[0].nodeValue;
var year=x[i].getElementsByTagName("year")[0].childNodes[0].nodeValue;
document.write("<br/>");
document.write("Book: "+(i+1)+"<br/>");
document.write("category: "+cat+"<br/>");
document.write("Title: "+title+"<br/>");
document.write("Author: "+author+"<br/>");
document.write("Year: "+year+"<br/>");
}
</script>
</body>
</html
```

In an organization 'X' Employees are categorised as :Technical employees, Non-Technical employees and Management employees, Every employee has FirstName, LastName, ContactNo and E-mailID as shown in the following sample data



Category of employee	FirstName	LastName	ContactNo	E-mailD
Technical	Bob	Dylan	1224567847	Bob@gmail.com
Non-technical	Gary	Moore	2546879546	Gary @gmail.com
Management	Rod	Stewart	8459621486	Rod @gmail.com

I. Create xml document for employees.

II. Display the result as per above table using XML DOM.

emp.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<employees>
<employee>
<category>Technical</category>
<FirstName>Bob</FirstName>
<LastName>Dylan</LastName>
<ContactNo>1234567</ContactNo>
<Email>Bob@gmail.com</Email>
</employee>
```

```xml
<employee>
<category>Non-Technical</category>
<FirstName>Gray</FirstName>
<LastName>Moore</LastName>
<ContactNo>25469855</ContactNo>
<Email>Gray@gmail.com</Email>
</employee>
<employee>
<category>Management</category>
<FirstName>Red</FirstName>
<LastName>Stewart</LastName>
<ContactNo>8455257257</ContactNo>
<Email>Red@gmail.com</Email>
</employee>
</employees>
```

Emp.html

```html
<!DOCTYPE html>
<html>
<style> table,th,td {
border : 1px solid black; border-collapse: collapse;
}
th,td { padding: 5px;
}
</style>
<body>
<table id="demo"></table>

<script>
    var xmlDoc;
if(window.XMLHttpRequest){
xmlDoc = new XMLHttpRequest();
}
else{
xmlDoc = new ActiveXObject("Microsoft.XMLHTTP");
xmlDoc.open("GET","emp.xml",false);
 xmlDoc.send();
var docxml = xmlDoc.responseXML;
var i;
var
table="<tr><th>Category</th><th>FirstName</th><th>LastName</th><th>ContactNo</
th><th>Email</th></tr>";
var x = docxml.getElementsByTagName("employee");
for (i = 0; i <x.length; i++)
```

```
{
table +=
"<tr><td>" + x[i].getElementsByTagName("category")[0].childNodes[0].nodeValue
+
"</td><td>" +
x[i].getElementsByTagName("FirstName")[0].childNodes[0].nodeValue +
"</td><td>" +
x[i].getElementsByTagName("LastName")[0].childNodes[0].nodeValue+
"</td><td>" +
x[i].getElementsByTagName("ContactNo")[0].childNodes[0].nodeValue+
"</td><td>" + x[i].getElementsByTagName("Email")[0].childNodes[0].nodeValue +
"</td></tr>";
}

document.getElementById("demo").innerHTML = table;
</script>
</body>
</html>
```

Assignment: 7

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="apply.xsl"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
  <cd>
    <title>Hide your heart</title>
    <artist>Bonnie Tyler</artist>
    <country>UK</country>
    <company>CBS Records</company>
    <price>9.90</price>
    <year>1988</year>
  </cd>
  <cd>
    <title>Greatest Hits</title>
    <artist>Dolly Parton</artist>
    <country>USA</country>
    <company>RCA</company>
    <price>9.90</price>
```

```xml
    <year>1982</year>
  </cd>
  <cd>
    <title>Still got the blues</title>
    <artist>Gary Moore</artist>
    <country>UK</country>
    <company>Virgin records</company>
    <price>10.20</price>
    <year>1990</year>
  </cd>
  <cd>
    <title>Eros</title>
    <artist>Eros Ramazzotti</artist>
    <country>EU</country>
    <company>BMG</company>
    <price>9.90</price>
    <year>1997</year>
  </cd>
  <cd>
    <title>One night only</title>
    <artist>Bee Gees</artist>
    <country>UK</country>
    <company>Polydor</company>
    <price>10.90</price>
    <year>1998</year>
  </cd>
  <cd>
    <title>Sylvias Mother</title>
    <artist>Dr.Hook</artist>
    <country>UK</country>
    <company>CBS</company>
    <price>8.10</price>
    <year>1973</year>
  </cd>
  <cd>
    <title>Maggie May</title>
    <artist>Rod Stewart</artist>
    <country>UK</country>
    <company>Pickwick</company>
    <price>8.50</price>
    <year>1990</year>
  </cd>
  <cd>
    <title>Romanza</title>
    <artist>Andrea Bocelli</artist>
    <country>EU</country>
    <company>Polydor</company>
    <price>10.80</price>
    <year>1996</year>
  </cd>
  <cd>
    <title>When a man loves a woman</title>
    <artist>Percy Sledge</artist>
    <country>USA</country>
    <company>Atlantic</company>
    <price>8.70</price>
    <year>1987</year>
  </cd>
  <cd>
    <title>Black angel</title>
    <artist>Savage Rose</artist>
    <country>EU</country>
    <company>Mega</company>
    <price>10.90</price>
    <year>1995</year>
  </cd>
  <cd>
    <title>1999 Grammy Nominees</title>
    <artist>Many</artist>
    <country>USA</country>
    <company>Grammy</company>
    <price>10.20</price>
    <year>1999</year>
  </cd>
  <cd>
    <title>For the good times</title>
    <artist>Kenny Rogers</artist>
    <country>UK</country>
    <company>Mucik Master</company>
    <price>8.70</price>
    <year>1995</year>
  </cd>
  <cd>
    <title>Big Willie style</title>
    <artist>Will Smith</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>9.90</price>
    <year>1997</year>
  </cd>
  <cd>
    <title>Tupelo Honey</title>
    <artist>Van Morrison</artist>
    <country>UK</country>
    <company>Polydor</company>
    <price>8.20</price>
    <year>1971</year>
  </cd>
  <cd>
    <title>Soulsville</title>
    <artist>Jorn Hoel</artist>
    <country>Norway</country>
    <company>WEA</company>
    <price>7.90</price>
    <year>1996</year>
  </cd>
  <cd>
    <title>The very best of</title>
    <artist>Cat Stevens</artist>
    <country>UK</country>
    <company>Island</company>
    <price>8.90</price>
    <year>1990</year>
  </cd>
  <cd>
    <title>Stop</title>
    <artist>Sam Brown</artist>
    <country>UK</country>
    <company>A and M</company>
    <price>8.90</price>
    <year>1988</year>
  </cd>
  <cd>
    <title>Bridge of Spies</title>
    <artist>T`Pau</artist>
    <country>UK</country>
    <company>Siren</company>
    <price>7.90</price>
    <year>1987</year>
  </cd>
  <cd>
    <title>Private Dancer</title>
    <artist>Tina Turner</artist>
    <country>UK</country>
    <company>Capitol</company>
    <price>8.90</price>
    <year>1983</year>
  </cd>
  <cd>
    <title>Midt om natten</title>
    <artist>Kim Larsen</artist>
    <country>EU</country>
    <company>Medley</company>
    <price>7.80</price>
    <year>1983</year>
  </cd>
  <cd>
    <title>Pavarotti Gala Concert</title>
    <artist>Luciano Pavarotti</artist>
    <country>UK</country>
    <company>DECCA</company>
    <price>9.90</price>
    <year>1991</year>
  </cd>
  <cd>
    <title>The dock of the bay</title>
    <artist>Otis Redding</artist>
    <country>USA</country>
    <company>Stax Records</company>
    <price>7.90</price>
    <year>1968</year>
  </cd>
  <cd>
    <title>Picture book</title>
    <artist>Simply Red</artist>
    <country>EU</country>
    <company>Elektra</company>
    <price>7.20</price>
    <year>1985</year>
  </cd>
  <cd>
    <title>Red</title>
    <artist>The Communards</artist>
    <country>UK</country>
    <company>London</company>
    <price>7.80</price>
    <year>1987</year>
  </cd>
  <cd>
    <title>Unchain my heart</title>
    <artist>Joe Cocker</artist>
    <country>USA</country>
    <company>EMI</company>
    <price>8.20</price>
    <year>1987</year>
  </cd>
</catalog>
```

1. write an xsl program to display contents in table, sort the artist, Filtering the artist=Bob Dylan for given xml document

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
          <table border="1">
            <tr bgcolor="#9acd32">
              <th>Title</th>
              <th>Artist</th>
            </tr>
            <xsl:for-each select="catalog/cd">
            <xsl:sort select="artist"/>
            <tr>
              <td><xsl:value-of select="title"/></td>
              <td><xsl:value-of select="artist"/></td>
            </tr>
            </xsl:for-each>
          </table>
        </body>
        </html>
      </xsl:template>

      </xsl:stylesheet>
```

29. Write an xsl program to implementation of xsl:if and xsl:choose element for given xml document

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
  <h2>My CD Collection</h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th>Title</th>
      <th>Artist</th>
      <th>Price</th>
    </tr>
    <xsl:for-each select="catalog/cd">
      <xsl:if test="price &gt; 10">
```

```
          <tr>
            <td><xsl:value-of select="title"/></td>
            <td><xsl:value-of select="artist"/></td>
            <td><xsl:value-of select="price"/></td>
          </tr>
        </xsl:if>
      </xsl:for-each>
    </table>
    </body>
    </html>
  </xsl:template>

  </xsl:stylesheet>
```

Choose:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
  <h2>My CD Collection</h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th>Title</th>
      <th>Artist</th>
      <th>Price</th>
    </tr>
    <xsl:for-each select="catalog/cd">
      <xsl:if test="price &gt; 10">
        <tr>
          <td><xsl:value-of select="title"/></td>
          <td><xsl:value-of select="artist"/></td>
          <td><xsl:value-of select="price"/></td>
        </tr>
      </xsl:if>
    </xsl:for-each>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

Assignment-8

Following table represents information of Bookstore with 4 sample records; each book has Category, Title, Author(one or more), Year and Price. Read the table carefully and make and implement the programs for the following:

- Create xml file for Bookstore.
- Validate XML document with XML Schema
- Display the contents of table as given in the following using Extensible Stylesheet Language Transformations (XSLT).

Category	Title	Author	Year	Price in rupees
Cooking	Everyday Italian	Giada De Laurentiis	2005	30.00
Children	Harry Potter	J K. Rowling	2005	29.99
Web1	XQuery Kick Start	James McGovern,Per	2003	49.50
Web2	Learning XML	Erik T. Ray	2003	39.00

.xml file

```xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="book.xsl"?>
<bookstore>
<book>
<category>cooking</category>
<title>Everyday Italian</title>
<author>Giada De Laurentiis</author>
<year>2005</year>
<price>30.00</price>
</book>
<book>
<category>children</category>
<title>Harry Potter</title>
<author>J K. Rowling</author>
<year>2005</year>
<price>29.99</price>
</book>
<book>
<category>web 1</category>
<title >XQuery Kick Start</title>
<author>James McGovern</author>
<author>Per Bothner</author>
<author>Kurt Cagle</author>
<author>James Linn</author>
```

```xml
<author>Vaidyanathan Nagarajan</author>
<year>2003</year>
<price>49.99</price>
</book>
<book >
<category>web 2</category>
<title>Learning XML</title>
<author>Erik T. Ray</author>
<year>2003</year>
<price>39.95</price>
</book>
</bookstore>
```

Book.xsd

```xml
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace = "http://bookstore.com"
elementFormDefault = "qualified"
>
 <xs:element name="bookstore">
 <xs:complexType>
 <xs:sequence>
 <xs:element name="book" maxOccurs="unbounded">
 <xs:complexType>
 <xs:sequence>
 <xs:element name="category" type="xs:string"/>
 <xs:element name="title" type="xs:string"/>
 <xs:element name="author" type="xs:string" maxOccurs="unbounded"/>
 <xs:element name="year" type="xs:integer"/>
 <xs:element name="price" type="xs:decimal"/>
 </xs:sequence>
 </xs:complexType>
 </xs:element>
 </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

Book.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
version="1.0"  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:template match="/">
 <html>
<style>
 table,th,td {
 border : 1px solid black;
 border-collapse: collapse;}
 th,td {padding: 5px;}
</style>
 <body>
 <h1>Bookstore Catalog</h1>
 <table>
 <tr>
 <th>Category</th>
 <th>Title</th>
 <th>Author</th>
 <th>Year</th>
 <th>Price</th>
 </tr>


<xsl:for-each select="bookstore/book">
<tr>

          <td><xsl:value-of select="category"/></td>
          <td><xsl:value-of select="title"/></td>
          <td><xsl:apply-templates select="author"/></td>
          <td><xsl:value-of select="year"/></td>
          <td><xsl:value-of select="price"/></td>
        </tr>
</xsl:for-each>

</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Apply

.xsl

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsl:stylesheet version = "1.0"
   xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">

   <xsl:template match = "/">
      <html>
         <body>
            <h2>Students</h2>
           <xsl:apply-templates/>
         </body>
      </html>
   </xsl:template>

   <xsl:template match = "class/student">
      <xsl:apply-templates select = "@rollno" />
   </xsl:template>

   <xsl:template match = "@rollno">
      <span style = "font-size = 22px;">
         <xsl:value-of select = "." />
      </span>
      <br />
   </xsl:template>

</xsl:stylesheet>
```

.xml

```
<?xml version = "1.0"?>
<?xml-stylesheet type = "text/xsl" href = "myroush.xsl"?>
<class>
   <student rollno = "393">
      <firstname>Dinkar</firstname>
      <lastname>Kad</lastname>
      <nickname>Dinkar</nickname>
      <marks>85</marks>
   </student>
   <student rollno = "493">
      <firstname>Vaneet</firstname>
      <lastname>Gupta</lastname>
      <nickname>Vinni</nickname>
      <marks>95</marks>
   </student>
   <student rollno = "593">
      <firstname>Jasvir</firstname>
      <lastname>Singh</lastname>
      <nickname>Jazz</nickname>
      <marks>90</marks>
   </student>
</class>
```

Restrictions on Values

The following example defines an element called "age" with a restriction. The value of age cannot be lower than 0 or greater than 120:

```
<xs:element name="age">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="120"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Restrictions on a Set of Values

To limit the content of an XML element to a set of acceptable values, we would use the enumeration constraint.

The example below defines an element called "car" with a restriction. The only acceptable values are: Audi, Golf, BMW:

```
<xs:element name="car">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Audi"/>
      <xs:enumeration value="Golf"/>
      <xs:enumeration value="BMW"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

The only acceptable value is ONE of the LOWERCASE letters from a to z:

```
<xs:pattern value="[a-z]"/>
```

The acceptable value is zero or more occurrences of lowercase letters from a to z: `<xs:pattern value="([a-z])*"/>`

The complexType element defines a complex type. A complex type element is an XML element that contains other elements and/or attributes.

The simpleType element defines a simple type and specifies the constraints and information about the values of attributes or text-only elements.

#REQUIRED: The attribute is required

#IMPLIED: The attribute is optional

#FIXED *value:* The attribute value is fixed

Well Formed XML

- XML Documents Must Have a Root Element
- The XML Prolog
- All XML Elements Must Have a Closing Tag
- XML Tags are Case Sensitive
- XML Elements Must be Properly Nested
- XML Attribute Values Must Always be Quoted
- Entity References
- Comments in XML
- White-space is Preserved in XML

XML Naming Rules

XML elements must follow these naming rules:

- Element names are case-sensitive
- Element names must start with a letter or underscore
- Element names cannot start with the letters xml (or XML, or Xml, etc)
- Element names can contain letters, digits, hyphens, underscores, and periods
- Element names cannot contain spaces

Any name can be used, no words are reserved (except xml).

My Favorite Way

The following three XML documents contain exactly the same information:

A date attribute is used in the first example:

```xml
<note date="2008-01-10">
  <to>Tove</to>
  <from>Jani</from>
</note>
```

A <date> element is used in the second example:

```xml
<note>
  <date>2008-01-10</date>
  <to>Tove</to>
  <from>Jani</from>
</note>
```

An expanded <date> element is used in the third example: (THIS IS MY FAVORITE):

```xml
<note>
  <date>
    <year>2008</year>
    <month>01</month>
    <day>10</day>
  </date>
  <to>Tove</to>
  <from>Jani</from>
</note>
```

Avoid XML Attributes?

Some things to consider when using attributes are:

- attributes cannot contain multiple values (elements can)
- attributes cannot contain tree structures (elements can)
- attributes are not easily expandable (for future changes)

XML Namespaces:

XML Namespaces provide a method to avoid element name conflicts.

Name Conflicts

In XML, element names are defined by the developer. This often results in a conflict when trying to mix XML documents from different XML applications.

This XML carries HTML table information:

```
<table>
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

This XML carries information about a table (a piece of furniture):

```
<table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

Solving the Name Conflict Using a Prefix

Name conflicts in XML can easily be avoided using a name prefix.

This XML carries information about an HTML table, and a piece of furniture:

```
<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>
```

```xml
<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

XML Namespaces - The xmlns Attribute

When using prefixes in XML, a **namespace** for the prefix must be defined.

The namespace can be defined by an **xmlns** attribute in the start tag of an element.

The namespace declaration has the following syntax. xmlns:*prefix*="*URI*".

```xml
<root>

<h:table xmlns:h="http://www.w3.org/TR/html4/">
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table xmlns:f="https://www.w3schools.com/furniture">
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>

</root>
```

Namespaces can also be declared in the XML root element:

```xml
<root xmlns:h="http://www.w3.org/TR/html4/"
xmlns:f="https://www.w3schools.com/furniture">
```

The purpose of using an URI is to give the namespace a unique name.

A **Uniform Resource Identifier** (URI) is a string of characters which identifies an Internet Resource.

The most common URI is the **Uniform Resource Locator** (URL) which identifies an Internet domain address

```
Default namespace
```

Namespaces in Real Use

XSLT is a language that can be used to transform XML documents into other formats.

The XML document below, is a document used to transform XML into HTML.

The namespace "http://www.w3.org/1999/XSL/Transform" identifies XSLT elements inside an HTML document.

Tip: If you want to style an XML document, use XSLT.

Why Does XML Display Like This?

XML documents do not carry information about how to display the data.

Since XML tags are "invented" by the author of the XML document, browsers do not know if a tag like <table> describes an HTML table or a dining table.

Without any information about how to display the data, the browsers can just display the XML document as it is.

The XMLHttpRequest Object

The XMLHttpRequest object can be used to request data from a web server.

The XMLHttpRequest object is **a developers dream**, because you can:

- Update a web page without reloading the page
- Request data from a server - after the page has loaded
- Receive data from a server - after the page has loaded
- Send data to a server - in the background.

```
Xml link

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

<homepages xmlns:xlink="http://www.w3.org/1999/xlink">
  <homepage xlink:type="simple" xlink:href="https://www.w3schools
.com">Visit W3Schools</homepage>
  <homepage xlink:type="simple" xlink:href="http://www.w3.org">Vi
sit W3C</homepage>
</homepages>
```

Change text :

```
 <h1 onclick="this.innerHTML = 'Ooops! text changed'">Click on this text! it
will change</h1>
```

- PCDATA will be treated as markup and entities will be expanded.
-