19CS2107 ENTERPRISE PROGRAMMING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SUBJECT CODE: 19CS2107

ENTERPRISE PROGRAMMING WORKBOOK

190031920 Nikhil Reddy Avuthu

XML #1

Pre-Lab Task:

1) What is the full form of XML?

Ans: XML – Extensible Markup Language.

2) What is the full form of DTD?

Ans: DTD – Document Type Definition.

3) What is the full form of XSD?

Ans: XSD – XML Schema Definition.

4)What are the rules to be followed to create well-formed XML documents?

Ans:

- All XML elements must have a closing tag.
- XML tags are case-sensitive.
- All XML elements must be properly nested.
- All XML documents must have a root element.
- Attribute values must always be quoted.
- With XML, whitespace is preserved.
- All element and attribute names contain either zero or one colon.
- No entity names, processing instruction targets, or notation names contain any colons.
- There must always be a prolog in beginning of an XML file.

5) Write down the functionalities and syntax of the below mentioned.

a) XML Naming Rules

Ans:

- 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).

b) XML Element

- An XML element is everything from (including) the element's start tag to (including) the element's end tag.
- An element can contain: Text, Attribute, Other elements, or a mix.
- There can also be empty XML elements.

Example:

<empty></empty>

Here,
<empty></empty> is an empty element.
Or it can be written as,
<element/> (Self closing tag)

c) XML attributes

- XML Attributes can have attributes just like HTML
- Attributes are designed to contain data related to a specific element.
- XML Attributes must be quoted.(Either single or double quotes can be used)

Example:

< devoleper name='NikhilReddy'>

Here**, <developer> is the element it has an attribute of name with the value 'NikhilReddy'.**

d) <!DOCTYPE>

- An XML document with correct syntax is called "Well Formed".
- An XML document validated against a DTD is both "Well Formed" and "Valid".
- The DOCTYPE declaration above contains a reference to a DTD file. The content of the DTD file is shown and explained below.

Example:

<!DOCTYPE note SYSTEM "Note.dtd">

The DOCTYPE declaration above contains a reference to a DTD file.

e) <!ELEMENT>

- In a DTD, elements are declared with an ELEMENT declaration.
- In a DTD, XML elements are declared with the following syntax:

  <!ELEMENT element-name category>

  Or

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

- In a DTD, elements can also be declared as empty elements.

  Syntax:

  <!ELEMENT element-name EMPTY>

Example:

**
**

f) <!ATTLIST>

- In a DTD, attributes are declared with an ATTLIST declaration.
- An attribute declaration has the following syntax:
  <!ATTLIST element-name attribute-name attribute-type attribute-value>
- There are different types of attributes.
- The **attribute-value** can be one of the following:

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

g) **simpleType**

- The simpleType element defines a simple type and specifies the constraints and information about the values of attributes or text-only elements.
- The **parent elements** are **attributes, elements, lists, restriction, schema, union.**
- Syntax:

<simpleType
id=ID
name=NCName
any attributes
>

(annotation?,(restriction|list|union))

</simpleType>

- (The ? sign declares that the element can occur zero or one time inside the simpleType element)

Example:

```
<xs:element name="age">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="100"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Here,

The above example defines an element called "age" that is a simple type with a restriction. The value of age can NOT be lower than 0 or greater than 100. **h) complexType:**

- The complexType element defines a complex type. A complex type element is an XML element that contains other elements and/or attributes.
- The **parent elements** are **elements, redefine, schema.**
- Syntax:

<complexType
id=ID
name=NCName
abstract=true|false mixed=true|false
block=(#all|list_of(extension|restriction)
)
final=(#all|list_of(extension|restriction))
any_attributes

>

(annotation?,(simpleContent|complexContent|((group|all|
choice|sequence)?,((attribute|attributeGroup)*,anyAttribute?))))

</complexType>

Example:

```
<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
  <xs:element name="to" type="xs:string"/>
  <xs:element name="from" type="xs:string"/>
  <xs:element name="heading" type="xs:string"/>
  <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

The above example has an element named "note" that is of a complex type.

4) Write down the functionalities of the below mentioned in DTD – Attributes

a)CDATA

CDATA means character data.

A CDATA section contains text that will NOT be parsed by a parser. Tags inside a CDATA section will NOT be treated as markup and entities will not be expanded. The primary purpose is for including material such as XML fragments, without needing to escape all the delimiters.

The only delimiter that is recognized in a CDATA section is "]]>" - which indicates the end of the CDATA section. CDATA sections cannot be nested.

b)PCDATA

PCDATA means parsed character data.

PCDATA is text that WILL be parsed by a parser. The text will be examined by the parser for entities and markup.

XML parsers are used to parse all the text in an XML document. PCDATA stands for Parsed Character data. PCDATA is the text that will be parsed by a parser. Tags inside the PCDATA will be treated as markup and entities will be expanded.

c)Default value

Example:

DTD:
<!ELEMENT square EMPTY>
<!ATTLIST square width CDATA "0">

Valid XML:
<square width="100" />

In the example above, the "square" element is defined to be an empty element with a "width" attribute of type CDATA. If no width is specified, it has a default value of 0.

d) #REQUIRED Syntax:

<!ATTLIST element-name attribute-name attribute-type #REQUIRED>

Example:
DTD:
<!ATTLIST person number CDATA #REQUIRED>

Valid XML:
<person number="5677" />

Invalid XML:
<person />

Use the #REQUIRED keyword if you don't have an option for a default value, but still want to force the attribute to be present.

e)#IMPLIED

Syntax:
<!ATTLIST element-name attribute-name attribute-type #IMPLIED>

DTD:
<!ATTLIST contact fax CDATA #IMPLIED> Valid
XML:
<contact fax="555-667788" /> Valid
XML:

<contact />

Use the #IMPLIED keyword if you don't want to force the author to include an attribute, and you don't have an option for a default value.

f)#FIXED value

Synatx:

<!ATTLIST element-name attribute-name attribute-type #FIXED "value">

DTD:
<!ATTLIST sender company CDATA #FIXED "Microsoft">

Valid XML:
<sender company="Microsoft" />

Invalid XML:
<sender company="XYZ" />

Use the #FIXED keyword when you want an attribute to have a fixed value without allowing the author to change it. If an author includes another value, the XML parser will return an error.

5) As you're in the very beginning stage of learning how to create a XML document.Write an XML to accept student details [Name, ID, Branch and CGPA] for minimum 5 students.

<class>
<student>
<name> ABC </name>
<id> 001 </id>
<branch> IT </branch>
<cgpa> 9 </cgpa>
</student>
<student>

CODE:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<class>
    <student>
        <name>Nikhil</name>
        <id>190031920</id>
        <branch>CSE</branch>
        <cgpa>9.8</cgpa>
    </student>
    <student>
        <name>Vyshnav</name>
        <id>190031927</id>
        <branch>CSE</branch>
        <cgpa>9.7</cgpa>
    </student>
    <student>
        <name>Goutham</name>
        <id>190031925</id>
        <branch>CSE</branch>
        <cgpa>9.6</cgpa>
    </student>
    <student>
        <name>anirudh</name>
        <id>190031956</id>
        <branch>CSE</branch>
        <cgpa>9.7</cgpa>
    </student>
    <student>
        <name>Mahesh</name>
        <id>190031999</id>
        <branch>CSE</branch>
        <cgpa>9.4</cgpa>
    </student>
</class>
```

In Lab Task:

a. Write a program for Books store, and the XML file is created that contains the information about five books and displaying the XML file using CSS.

CODE:

XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="styles.css"?>
<books>
    <heading>Welcome To Enterprise programming</heading>
    <book>
        <title>Title -: Web Programming</title>
        <author>Author -: Chrisbates</author>
        <publisher>Publisher -: Wiley</publisher>
        <edition>Edition -: 3</edition>
        <price> Price -: 300</price>
    </book>
    <book>
        <title>Title -: Internet world-wide-web</title>
        <author>Author -: Ditel</author>
        <publisher>Publisher -: Pearson</publisher>
        <edition>Edition -: 3</edition>
        <price>Price -: 400</price>
    </book>
    <book>
        <title>Title -: Computer Networks</title>
        <author>Author -: Foruouzan</author>
        <publisher>Publisher -: Mc Graw Hill</publisher>
        <edition>Edition -: 5</edition>
        <price>Price -: 700</price>
    </book>
    <book>
        <title>Title -: DBMS Concepts</title>
```

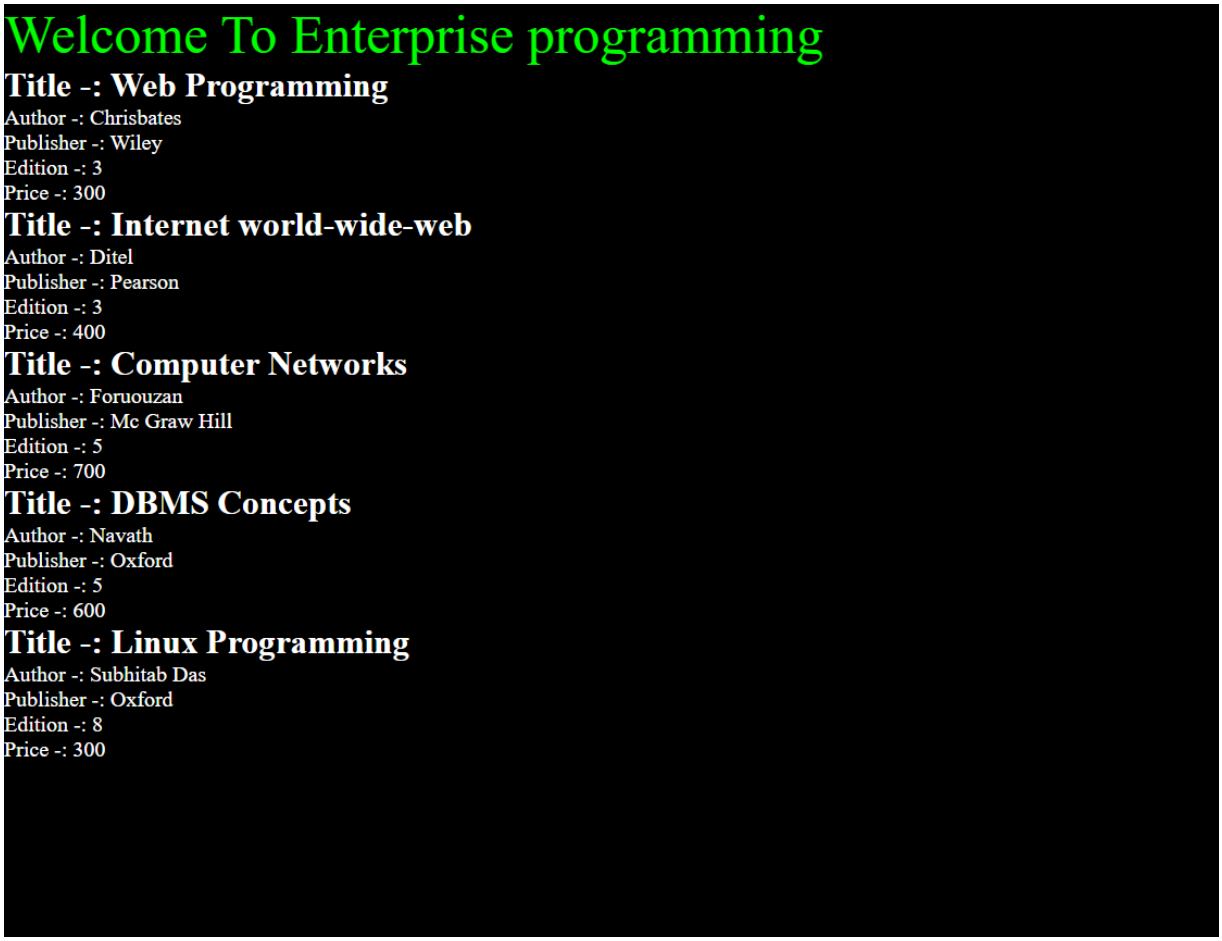
```
        <author>Author -: Navath</author>
        <publisher>Publisher -: Oxford</publisher>
        <edition>Edition -: 5</edition>
        <price>Price -: 600</price>
    </book>
    <book>
        <title>Title -: Linux Programming</title>
        <author>Author -: Subhitab Das</author>
        <publisher>Publisher -: Oxford</publisher>
        <edition>Edition -: 8</edition>
        <price>Price -: 300</price>
    </book>
</books>
```

CSS:

```
books {
    color: white;
    background-color: rgb(0, 0, 0);
    width: 100%;
}
heading {
    color: rgb(0, 255, 0);
    font-size: 40px;
    background-color: rgb(0, 0, 0);
}
heading,
title,
author,
publisher,
edition,
price {
    display: block;
```

```
}
title {
    font-size: 25px;
    font-weight: bold;
}
```

OUTPUT:



Post Lab Task:

Write a program for Books store, and the XML file is created that contains the information about five books of different categories and displaying the XML file using CSS.

Action and **Adventure**.
Classics.
Comic Book or **Graphic Novel**.

Detective and Mystery.
Fantasy.
Historical Fiction.
Horror.
Literary Fiction.

CODE:

XML:

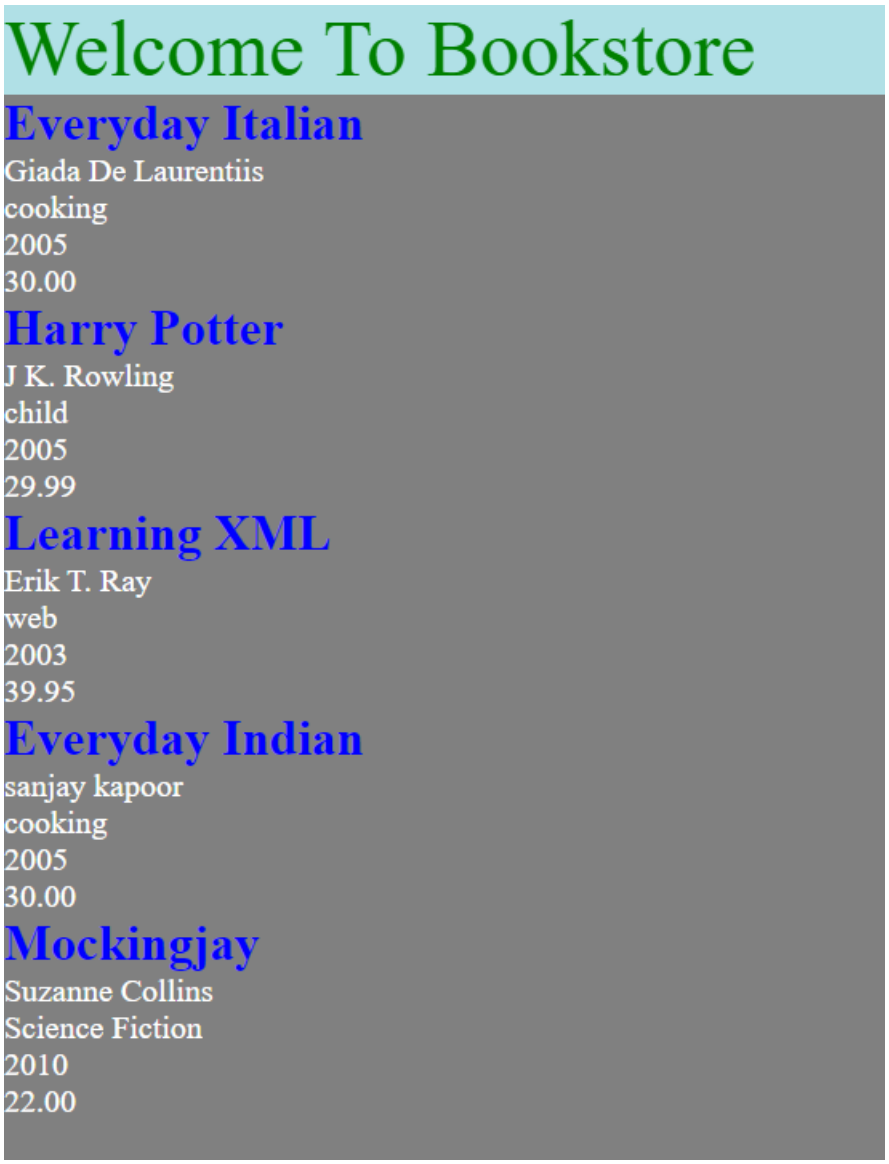
```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="postlab1styles.css"?>
<bookstore>
    <heading>Welcome To Bookstore</heading>
    <book>
        <title lang="en">Everyday Italian</title>
        <author>Giada De Laurentiis</author>
        <category>cooking</category>
        <year>2005</year>
        <price>30.00</price>
    </book>
    <book>
        <title lang="en">Harry Potter</title>
        <author>J K. Rowling</author>
        <category>child</category>
        <year>2005</year>
        <price>29.99</price>
    </book>
    <book>
        <title lang="en">Learning XML</title>
        <author>Erik T. Ray</author>
        <category>web</category>
        <year>2003</year>
        <price>39.95</price>
    </book>
    <book>
        <title lang="en">Everyday Indian</title>
        <author>sanjay kapoor</author>
        <category>cooking</category>
        <year>2005</year>
```

```
        <price>30.00</price>
    </book>
    <book>
        <title lang="en">Mockingjay</title>
        <author>Suzanne Collins</author>
        <category>Science Fiction</category>
        <year>2010</year>
        <price>22.00</price>
    </book>
</bookstore>
```

CSS:

```
bookstore {
    color: white;
    background-color: gray;
    width: 100%;
}
heading {
    color: green;
    font-size: 40px;
    background-color: powderblue;
}
heading,
title,
author,
category,
year,
price {
    display: block;
}
title {
    font-size: 25px;
    color: blue;
    font-weight: bold;
}
```

OUTPUT:


Welcome To Bookstore
Everyday Italian
Giada De Laurentiis
cooking
2005
30.00
Harry Potter
J K. Rowling
child
2005
29.99
Learning XML
Erik T. Ray
web
2003
39.95
Everyday Indian
sanjay kapoor
cooking
2005
30.00
Mockingjay
Suzanne Collins
Science Fiction
2010
22.00