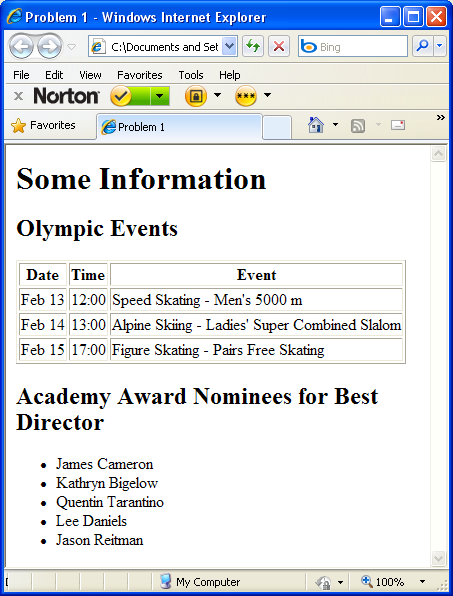
44-563 Web Services Technology Exam 1 Spring 2010

1. (12 pts) Complete the html code for this web page. You may continue onto the back of this page.



**<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">**

**<html>**

**<head>**

**<title>**

**Problem 1**

**</title>**

**<meta http-equiv="Content-Type"**

**content="text/html; charset=UTF-8">**

**</head>**

**<body>**

**<h1>Some Information</h1>**

**<h2>Olympic Events</h2>**

**<table border="1">**

**<thead>**

**<tr>**

**<th>Date</th>**

**<th>Time</th>**

**<th>Event</th>**

**</tr>**

**</thead>**

**<tbody>**

**<tr>**

**<td>Feb 13</td>**

**<td>12:00</td>**

**<td>Speed Skating - Men's 5000 m</td>**

**</tr>**

**<tr>**

**<td>Feb 14</td>**

**<td>13:00</td>**

**<td>Alpine Skiing - Ladies' Super Combined Slalom</td>**

**</tr>**

**<tr>**

**<td>Feb 15</td>**

**<td>17:00</td>**

**<td>Figure Skating - Pairs Free Skating</td>**

**</tr>**

**</tbody>**

**</table>**

**<h2>Academy Award Nominees for Best Director</h2>**

**<ul>**

**<li>James Cameron</li>**

**<li>Kathryn Bigelow</li>**

**<li>Quentin Tarantino</li>**

**<li>Lee Daniels</li>**

**<li>Jason Reitman</li>**

**</ul>**

**</body>**

**</html>**

2. (8 pts) For the web page shown in Problem 1, write the code that would be included in a cascading style sheet to

(a) display the directors' names in italic letters

**li {**

**font-style: italic;**

**}**

(b) display the "Some Information" heading in red letters

**h1 {**

**color: red;**

**}**

(c) set the background color of the page to blue

**body {**

**background-color: blue;**

**}**

(d) set the background color of the table to yellow

**table {**

**background-color: yellow;**

**}**

3. (10 pts) The student language is defined by this DTD file, named **student.dtd**.

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

**<!--**

**Document : student.dtd**

**Description: Describes the student language**

**-->**

**<!ELEMENT student (name, courseList)>**

**<!ATTLIST student studentNumber ID #REQUIRED>**

**<!ELEMENT name (first, last)>**

**<!ELEMENT first (#PCDATA)>**

**<!ELEMENT last (#PCDATA)>**

**<!ELEMENT courseList (course+)>**

**<!ELEMENT course (courseNumber, grade)>**

**<!ELEMENT courseNumber (#PCDATA)>**

**<!ELEMENT grade (#PCDATA)>**

Write an XML file **student.xml** that uses the student language to represent the data described below.

Leonard Poodle is a student with student number S123654. He is taking two courses: 99‑888 with a grade of 93 and 11-222 with a grade of 86.

**<student studentNumber="S123456">**

**<name>**

**<first>Leonard</first>**

**<last>Poodle</last>**

**</name>**

**<courseList>**

**<course>**

**<courseNumber>99-888</courseNumber>**

**<grade>93</grade>**

**</course>**

**<course>**

**<courseNumber>11-222</courseNumber>**

**<grade>86</grade>**

**</course>**

**</courseList>**

**</student>**

4. (a) (2 pts) Each namespace has a unique URI.

TRUE FALSE

(b) (2 pts) Each namespace has a unique prefix that must be used in all documents that use the namespace.

TRUE FALSE

(c) (2 pts) In the namespace declaration below

**xmlns:mml="http://www.w3.org/1998/Math/MathML"**

the namespace prefix is \_\_\_\_\_\_mml\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

and the namespace URI is \_http://www.w3.org/1998/Math/MathML\_

5. (10 pts) Assume we want to define a language to represent information about cities. Here is a sample XML file that uses the language.

**<!DOCTYPE cityList SYSTEM 'cities.dtd'>**

**<cityList>**

**<city name="Dallas">**

**<population>5000000</population>**

**<temperature>**

**<avrgHigh>82</avrgHigh>**

**<avrgLow>68</avrgLow>**

**</temperature>**

**</city>**

**<city name="Detroit">**

**<population>9000000</population>**

**<temperature>**

**<avrgHigh>68</avrgHigh>**

**<avrgLow>42</avrgLow>**

**</temperature>**

**</city>**

**</cityList>**

Write the DTD file cities.dtd that defines this language. The XML file above must validate against the DTD you write. The city name is required and must be unique.

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

**<!ELEMENT cityList (city+)>**

**<!ELEMENT city (population, temperature)>**

**<!ATTLIST city name ID #REQUIRED>**

**<!ELEMENT population (#PCDATA)>**

**<!ELEMENT temperature (avrgHigh, avrgLow)>**

**<!ELEMENT avrgHigh (#PCDATA)>**

**<!ELEMENT avrgLow (#PCDATA)>**

6. (10 pts) Assume an XML file has the following format:

**<cityList>**

**<city>**

**<name>Dallas</name>**

**<population>5000000</population>**

**<temperature>**

**<avrgHigh>82</avrgHigh>**

**<avrgLow>68</avrgLow>**

**</temperature>**

**</city>**

*\*\*\*\*\* more city elements \*\*\*\*\**

**</cityList>**

We want to use XSL to transform this data into a table in HTML. The list will contain the name and avrgHigh value for each city with a population greater than 250000. Write XSL code that produces this list from the data file. **Write ONLY the code that goes between the <table> and </table> tags. Everything else can be omitted.**

**<table>**

**<xsl:for-each select="cityList/city">**

**<xsl:if test="population &gt; 250000">**

**<tr>**

**<td>**

**<xsl:value-of select="name"/>**

**</td>**

**<td>**

**<xsl:value-of select="temperature/avrgHigh"/>**

**</td>**

**</tr>**

**</xsl:if>**

**</xsl:for-each>**

**</table>**

7. (10 pts) Answer the questions below concerning this definition from a schema:

**<xsd:complexType name="Items">**

**<xsd:sequence>**

**<xsd:element name="item" minOccurs="1" maxOccurs="10">**

**<xsd:complexType>**

**<xsd:sequence>**

**<xsd:element name="productName" type="xsd:string"/>**

**<xsd:element name="quantity">**

**<xsd:simpleType>**

**<xsd:restriction base="xsd:positiveInteger">**

**<xsd:maxExclusive value="12"/>**

**</xsd:restriction>**

**</xsd:simpleType>**

**</xsd:element>**

**<xsd:element name="USPrice" type="xsd:decimal"/>**

**<xsd:element ref="comment" minOccurs="0"/>**

**<xsd:element name="shipDate" type="xsd:date" minOccurs="1"/>**

**</xsd:sequence>**

**<xsd:attribute name="partNum" type="SKU" use="required"/>**

**</xsd:complexType>**

**</xsd:element>**

**</xsd:sequence>**

**</xsd:complexType>**

**<xsd:simpleType name="SKU">**

**<xsd:restriction base="xsd:string">**

**<xsd:pattern value="\d{2}[A-Z]{2}\d{4}"/>**

**</xsd:restriction>**

**</xsd:simpleType>**

(a) The **item** element can be omitted.

TRUE FALSE

(b) The **item** element can occur 20 times.

TRUE FALSE

(c) The **shipDate** element is required.

TRUE FALSE

(d) The value of **quantity** can be 0.

TRUE FALSE

(e) The maximum legal value of **quantity** is \_\_\_\_11\_\_\_\_\_\_.

(f) An **item** has an attribute named \_\_partNum\_\_\_\_\_.

(g) Give an example of a legal value for **partNum**. 38HP7249

8. Assume we are using Java to parse the xml document below.

**<dogList>**

**<dog>**

**<dogName>Tex</dogName>**

**<breed>Standard Poodle</breed>**

**<age>6</age>**

**<owner>**

**<ownerName>**

**<first>Mary</first>**

**<last>Smith</last>**

**</ownerName>**

**<address>**

**<city>Dallas</city>**

**<state>Texas</state>**

**</address>**

**</owner>**

**</dog>**

**REMAINING dog ELEMENTS NOT SHOWN TO SAVE SPACE**

**</dogList>**

(a) (3 pts) In order to extract the breed of the third dog, we would use the expression (*write the appropriate value for the first argument* ***below*** *the method call*):

**path.evaluate(" ", doc)**

**"/dogList/dog[3]/breed"**

(b) (3 pts) In order to extract the first name of the owner of the fifth dog, we would use the expression:

**path.evaluate(" ", doc)**

**"/dogList/dog[5]/owner/ownerName/first"**

(c) (3 pts) In order to extract the city of the owner of the first dog, we would use the expression:

**path.evaluate(" ", doc)**

**"/dogList/dog[1]/owner/address/city"**