XML Schema Lab Exercises

Lab 1. Converting an Existing DTD to an XML Schema

For this lab you are to convert juicers.dtd into an

equivalent XML Schema[1].

```
<!ELEMENT juicers (juicer)*>
<!ELEMENT juicer (name, image, description, warranty?, weight?,
cost+, retailer)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT image (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT warranty (#PCDATA)>
<!ELEMENT weight (#PCDATA)>
<!ELEMENT cost (#PCDATA)>
<!ELEMENT retailer (#PCDATA)>
```

Note the following DTD:

ELEMENT indicates element
* indicates zero or more
? indicates zero or one
+ indicates one or more
PCDATA indicates string

Do a straight one-to-one conversion from the above DTD to XML Schema Once you have the schema completed, validate it. Then modify juicers.xml to indicate that it conforms to your schema. Finally, validate the instance document against your schema. [1] It is a simplified version of the juicers.dtd. Namely, it doesn't contain any attribute declarations.

Lab 1.1 Default XMLSchema Namespace

For this lab you are to rework lab 1 so that XMLSchema is the default namespace. After you have done this validate your instance document against your new schema.

Lab 2. Using Embedded Element Declarations

For this lab you are to rework the juicers schema that you created in lab 1. You are to make it more compact. That is, rather than declaring an element and then ref'ing it, you are to inline the actual element declaration. Validate the instance document against your schema.

Lab 3. Using the Built-in Datatypes

For this lab you are to rework the juicers.xsd that you created in Lab 2. You are to change the datatypes from string to another built-in datatype, where it makes sense. Hint: weight, cost, and retailer can be declared with a much more relevant datatype than string. As always, validate the instance document against your schema.

Lab 4. Creating a New Datatype

Modify the schema that you created in Lab 3. Create a new datatype called money and declare the juicer cost element to be of that type. As always, validate the instance document against your schema.

Hint: here are the facets for the decimal built-in datatype:

totalDigits - the total number of digits allowable in the number
(including the digits to the right of the decimal point)
fractionDigits - the number of digits allowed to the right of the decimal point
pattern
enumeration
whitespace
maxInclusive
maxExclusive
minInclusive
minExclusive

Lab 5. Using Regular Expressions

Modify the schema that you created in Lab 4. Create a new datatype called imageType and declare the image element to be of that type. You are to define a Regular Expression for imageType. As always, validate the instance document against your schema.

Lab 6. Using Derived Types

Modify the schema that you created in Lab 5. Create a type called appliance. Define appliance to contain declarations for description and warranty. Create a type called juiceAppliance, which is derived from appliance (by extension). Make juicer of type juiceAppliance. As always, validate the instance document against your schema.

Note: You will need to rearrange the contents of each juicer in the instance document!

Lab 7. Creating a SubstitutionGroup

Modify the schema that you created in Lab 6. "warranty" and "guarantee" are terms that are used interchangeably in our culture (U.S.). Thus, our schema should reflect that by allowing the writer of an XML instance document to use either the tag <warranty> or <guarantee>. For this lab you are to create a substitution group containing warranty and guarantee. Then, modify juicers.xml such that sometimes <warranty> is used and sometimes <guarantee> is used. As always, validate the instance document against your schema.

Lab 8.a Using Attributes

In this directory you will find a version of juicers.xml which uses attributes. I have included juicers.dtd to show how the attributes were declared. Modify the schema that you created in Lab 7 to incorporate these attributes. As always, validate the instance document against your schema.

Lab 8.b Using Attributes (cont.)

I have enhanced juicers.xml and juicers.dtd to have a currency attribute for cost. Modify the schema that you created in Lab 8.a to incorporate this attribute. As always, validate the instance document against your schema.

Lab 8.c Using Attributes (concluded)

I have further enhanced juicers.xml and juicers.dtd to have a units attribute for weight. Modify the schema that you created in Lab 8.b to incorporate this attribute. As always, validate the instance document against your schema.

Hint: You can specify an attribute to have a fixed value as follows:

```
    <attribute       name="speed"       type="positiveInteger"
fixed="200"/>
```

Lab 9. Elements in Any Order

There is no inherent reason that the children of the juicer element should occur in a specified order. Let's change the schema so that the juicer element contains the same set of child elements but allow them to occur in any order. Once you've done that, shuffle the elements around in juicers.xml. As always, validate the instance document against your schema.

Notes:

[1] Whenever a type extends a base type, the extension elements are always appended to the base type's elements:

b1, b2, e1, e2

where b1 and b2 are the base type's elements and e1, e2 are the elements from the type that is extending the base type.

Thus, even if you make the base type unordered and the type extending the base type unordered, the resulting set of elements will still be partially ordered:

{b1, b2}, {e1, e2}

where the curly braces indicates an unordered set. We have an unordered first set followed (sequentially) by an unordered second set.

So, for this lab you are to delete the appliance type and place its element declarations in with the juiceAppliance type.

[2] One of the restrictions on using the <all> element is that the elements declared within <all> must have a maxOccurs="1". So, remove the cost that specifies the cost in Canadian Dollars. (We will just deal in US currency.)

Lab 10. Using Empty

Revert back to the schema that you created in lab 8.c (where you had an appliance complexType).

Currently the retailer element has as its content a URI. Modify it so that its content is empty and it instead has an attribute - href - whose type is anyURI. Once you've done that, modify juicers.xml to reflect this change. As always, validate the instance document against your schema.

Lab 11.a Creating an Appliances Repository Schema

For this lab you are to modify the juicers.xml and juicers.xsd that you created in the last lab (lab 10).

In all of the labs thus far we have stored all of our schema components in a single schema. Typically, however, you will put elements/types that may be used by many schemas into its own schema (a repository schema). In our juicer example the appliance type is an example of something that could be used in many schemas. Hence, let's pull it out of juicers.xsd and put it into a generic schema, appliances.xsd. Then modify juicers.xsd to use the definition of appliances in appliances.xsd. For this lab you are to use the same namespace for the two schemas.

Lab 11.b Demonstating the Chameleon Effect

For this lab you are to modify the Appliances.xsd schema that you created in the last lab (lab 11.a). Modify the Appliances.xsd schema to have no targetNamespace. Validate juicers.xml.Lab

11.c Using Multiple Namespaces

For this lab you are to modify the juicers.xml and the schemas that you created in lab 11.a.

Put Appliances.xsd schema in its own namespace (http://www.appliances.org). Modify juicers.xsd to use the appliances complexType, which is now in a different namespace. You will need to modify the instance document to reflect the fact that it is using multiple namespaces.Lab

11.d Creating Lists

For this lab you are to modify the juicers.xml and the schemas that you created in lab 11.c.

There are multiple retailers for each juicer. Modify juicers.xsd to allow for a list of values for the href in the retailers element.

Lab 12. Making the Schema Dynamically Evolvable

Currently the schema is very static. Instance documents must rigidly conform to the schema. The only avenue for evolution is to change the schema. When the schema was formed by a committee that will entail a lot of effort and will be a slow process - not satisfactory in a quickly changing marketplace.

The <any> element facilitates rapid evolution of the schema. For this lab you are to enable the juicer instance documents to grow at the bottom of each <juicer> element's content model. In this directory I have provided a schema, SchemaRepository.xsd. It contains an element, color. You are to extend your juicers document with this new element. Use your schema from lab11.d as your starting point. As always, validate the instance document against your schema.

Lab 12.1 Making the Juicer Information Net-Centric

Up till now the juicers data was all in one, monolithic XML document. This is not net-centricity! I have split up the data. Now there is an XML document for each juicer.

You are to:

1. Create juicers.xml

  - this XML document should be designed to link to each juicer document

   Notes:

    use physical URLs, since we don't have a web server running I have already created a skeletal version of juicers.xml for you

2. Create juicers.xsd

3. Create juicer.xsd

Use the schema from Lab10 as your starting point. I have provided XLink.xsd for you in this folder. You can use it. Validate each juicer document, and juicers.xml

Lab 13. Creating a Schema for your Home Appliances

Create a schema called homeAppliances.xsd. The schema should be designed to enable an instance document to list all the appliances in a home, e.g.,

```
<homeAppliances>
   <appliance xsi:type="j:juiceAppliance">
      ...
   </appliance>
   <appliance xsi:type="s:stereoAppliance">
      ...
   </appliance>
</homeAppliances>
```

The contents of each home appliance will be:

juicer appliance: elements applicable to all appliances (i.e., use appliances.xsd) plus name, image, weight, cost, retailer.

Hint: Work smart, not hard!!! Reuse juicers.xsd as well as appliances.xsd (from lab11.d)!!!

stereo appliance: elements applicable to all appliances (i.e., use appliances.xsd) plus brand and type.
As always, validate the instance document against your schema.

Lab 14. Creating a Schema for your Home Appliances, Version 2

After completing lab 13 you may have felt dissatisfied with the end result. Let's consider its deficiencies:

[1] We made the element appliance of type a:appliance. Thus, in the instance document one of our home appliances could have been an appliance. appliance really represents a class of things, and should never be an instance. The instances should be juiceAppliance, stereoAppliance, etc. (Analogously, it wouldn't make sense to allow an instance called person. Instead, the instances should be Roger, Sally, Pete, etc.)

[2] Instead of writing our instance document as:

```
<homeAppliances>
    <appliance xsi:type="j:juiceApplicance">
       ...
    </appliance>
    <appliance xsi:type="stereoApplicance">
       ...
    </appliance>
</homeAppliances>
```

It would be much more readable if we could write it as:

```
<homeAppliances>
    <juicer>
       ...
    </juicer>
    <stereo>
       ...
    </stereo>
 </homeAppliances>
```

Modify your work from lab 13 so that it no longer has these deficiencies. As always, validate the instance document against your schema.