Lab #10
XML and XSLT
Due in Lab, December 1, 2004

Name: _____

Lab Time: _____

Grade: _____/10

In this lab, you will gain experience working with XML and XSLT. XML is a language for creating markup languages. A markup language is any language that includes both text and "structure." The structure typically comes from enclosing text within tags. For instance, in the Hypertext Markup Language (HTML), if you want to make a phrase appear in **bold** font in a web browser you use the "b" tag: Lab 10 is **very** important.

As we discussed in lecture, a problem with HTML is that it is too closely tied to presentation issues and, thus, does not do a good job conveying semantic information about a document. Thus, on CNN's home page, a reader determines what text represents a "headline" and what text represents supporting text by formatting alone. The actual HTML does not explicitly state that something is a headline by using a headline tag. XML is designed to address this problem. It allows a developer to create a language with tags that help to encode a document's structure more explicitly and help to keep this structure separate from presentation issues. Thus, you could explicitly use a headline tag in an XML document and worry about its appearance later.

One technology for dealing with the presentation of XML documents is the XML Stylesheet Language, in particular, its transformation component, or XSLT. The idea is that XSLT is used to translate an XML document into HTML, where it is displayed to a user via a Web browser. However, XSLT is more generic than just an XML→HTML translator. It can be used to translate XML into multiple formats, such as XML (i.e. translate an XML document using one DTD into an XML document using a different DTD), PDF, MS Word, etc.

In this lab, you will gain experience with each of these steps. You will create a document type definition (also known as a DTD), write an XML document that uses that DTD, and translate that document into an HTML file using XSLT. In order to use XSLT, you will be writing an XSLT stylesheet to do the translation. Therefore, when you complete this lab, you should upload three documents: your XML file, your XSLT stylesheet, and the HTML file that is generated by your stylesheet.

Writing the Document Type Definition

In this section, we describe a document structure that you must encode into an XML document type definition. In particular, your goal is to store

people's contact information such as their names, phone numbers, e-mail, Web, and postal addresses. The root element of the document is **addressbook**. An addressbook can contain zero or more **address** elements. An address element contains elements for a **name**, **phone**, an optional **e-mail** address, an optional Web **homepage** URL, and a **postal address**. All of these elements contain just text except for the name and the postal address. A name consists of a **first name**, an optional **middle initial**, and a **last name**. A postal address consists of a **street**, an optional **apartment**, a **city**, a **state**, and a **zipcode**.

A document type definition appears in the DOCTYPE tag at the beginning of an XML document. Your document type definition should appear in your XML file at the top using the following template as a guide.

```
<?xml version="1.0"?>
<!DOCTYPE addressbook [
...DTD Goes Here...
]>
<addressbook>
...Data Goes Here...
</addressbook>
```

Create an xml file called lab10.xml. Type the template listed above into the file and replace the line that says "...DTD Goes Here..." with your DTD to contain all of the elements listed above, e.g. name, phone number, etc. Look at the lecture notes for the syntax rules of the DTD.

The Address Data

Now that you have written a DTD, you need some data that follows your DTD's rules. Encode the following information into your XML file (replace the line that says "...Data Goes Here..."), using the tags defined by your DTD.

```
Dave Williams
62000 Pearl Street
Boulder, CO 80302
(555) 232-4567
dave@dave.net

Kevin G. Andrews
2030 Tiverton Avenue
Apt.  368
Los Angeles CA, 90024
(555) 826-2321
<http://www.abode.com/home/kevin/>
kevin@abode.com
```

```
Jessica James
17390 Graham Lane
Santa Clara, CA 95050-4037
(555) 343-5781
jj@cscw.com

Janie Blair
3 Corte Playa Corona
San Diego CA, 92124-4134
(555) 324-5567

Cynthia J. Mulvihill
37245 Jupiter Hills Court
Unit 56
Arnold, GA 21012
(555) 223-0967
<http://www.notanaddress.com/users/cynthia/>

Jill Covert
24350 Brad Street
San Antonio TX 92115
(555) 547-3348
jill@redbull.com

Kathy Anderson
7314 Oak Drive
Riverside, CA, 92506
(555) 684-6249
kathy@e-biz.com
```

Verifying Your File

In this section, you will test that your XML file is valid. You will do this by passing your file to a shell script that will invoke an XML parser on your file. Use the following command:

```
~csci3308/bin/validate lab10.xml
```

. The validate script invokes the Java virtual machine which is located in the directory `/tools/cs/j2sdk/bin/`.

If the parser encounters any errors with your file, it will stop and print an error message. Otherwise, it will print out a line that looks like this:

```
lab10.xml:  197 ms (82 elems, 0 attrs, 1022 spaces, 598 chars)
```

Actually, the critical information in the line above is the "82 elems, 0 attrs". While your XML file may have different spaces which can lead to different numbers for the "spaces" and "chars" counts, your file when complete should

contain 82 elements to encode the address information above. Keep working on your XML file until the shell script indicates that your XML file is valid and that it contains 82 elements.

Writing the Stylesheet

You will now translate your XML file into an HTML file. Your goal is to take your XML file and convert it into the HTML file located at:

```
http://www.cs.colorado.edu/users/kena/lab10.html
```

Load this file into a Web browser to see your goal. Your stylesheet must create a table that is sorted by last name. Names are displayed in bold, and e-mail and Web addresses are presented in italics.

Create a file called lab10.xsl and type the following template into it.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
...Template Rules Go Here...
</xsl:stylesheet>
```

Replace the line that says "...Template Rules Go Here..." with the XSLT rules that will recreate the HTML file listed above. Take a look at lecture 29 for an example XSLT stylesheet. To create your HTML file, you will use the following command line:

```
~csci3308/bin/translate lab10.xml lab10.xsl
```

This shell script will create a file called lab10.html (if there are no errors in your stylesheet) otherwise it will display an error message. Use the error messages to debug your stylesheet.

Iterate on your stylesheet. Add a template rule and run the translate script to see the results. Do not try to write the whole stylesheet at once. Develop it in steps until you have created a page that looks like the one at the above URL.

Note: If you need to add whitespace to the output being created by your stylesheet, you can use the XSLT tag "text". i.e. if you want to add a space, followed by a hyphen, followed by a space to the output, add the following to your XSLT stylesheet template rule:

```
<xsl:text> - </xsl:text>
```

Note: If you need to add an HTML tag to your output that does not normally require a close tag, such as the "br" element. You can use the XSLT "element" tag, like this:

```
<xsl:element name="br" />
```

When you are finished upload your XML file, your XSLT stylesheet, and your generated HTML file into the moodle. To do so, create a tar file called <lastname>-lab10.tar.