Project 2

XTube

139 Points

due by 12:30 P.M. ET on Thursday, 16 March 2006

It is recommended that you read the entirety of this document,
well in advance of this project's deadline, before answering any of its questions.

It is further recommend that you begin answering this document's questions immediately thereafter.
Believe us this time.

Goals.

The goals of this project are to:

- Challenge you to solve a number of real-world business problems.
- Give you hands-on experience with SVG, XPath, XSL-FO, and XSLT.
- Introduce you to Apache's Xalan-J 2.7.0, an industry-standard XSL processor, to Apache's FOP 0.20.5, an XSL-FO processor, and to Adobe SVG Viewer 3.0x, an SVG viewer.

Grading Metric.

Each question is worth the number of points specified parenthetically in line with it.

Your responses to questions requiring exposition will be graded on the basis of their clarity and correctness. Your responses to questions requiring code will be graded on the following bases.[1]

Basis	Considerations
Correctness	Does your code work in accordance with the question's guidelines?
Design	Does your code make sense, given the question's framework? Is your code written logically, clearly, and succinctly? Is your code efficient? Is your code divided into logical units (*e.g.*, multiple templates)?
Style	Is your code rigorously documented with inline comments? Is it clear from your comments alone how your code operates? Is your code pretty-printed? Are your attributes, elements, templates, and variables aptly named?

Academic Honesty.

If, in the interests of development and debugging, you opt to publish XSLT-generated SVG or XHTML on `www.people.fas.harvard.edu`, it is expected that you will ensure the privacy of your work by password-protecting any Web-accessible directories and naming files in such a (random) way as to render their viewing on `nice.fas.harvard.edu` unlikely.

Recall that, for Project 1, you configured your account with a `~/public_html/` directory and a password-protected `~/public_html/cscie259/` directory.

Needless to say, attempting to view the work of another student, even if published in a world-accessible directory, is considered academic dishonesty and will be handled accordingly.

[1] By code, we mean SVG, XHTML, XML, XPath, XSL-FO, and XSLT.

Getting Started.

1. (0 points.) Damn, another 0-point question.

2. (0 points.) If you intend to do your work on `nice.fas.harvard.edu`, SSH to said machine and execute the following sequence of commands.[2,3]

```
cp -r ~cscie259/pub/distribution/projects/project2-6.0/ ~/cscie259/
cd ~/cscie259/
ls
```

You should see that you have the following in your current working directory

```
project1-6.0/       project2-6.0/
```

If, on the other hand, you do not intend to do your work on `nice.fas.harvard.edu`, you may proceed to download a `gzip`-compressed tarball or a ZIP file containing this `project2-6.0/` directory from the course's website to your local machine. Or, of course, can you transfer the directory itself via SFTP to your local machine. However, prior to disconnecting from `nice.fas.harvard.edu`, be sure to examine `cscie259.cshrc` in `~cscie259/pub/etc/`, so that you know how to configure your machine. Note that we expect use of J2SE SDK 1.4.2, Apache's Ant 1.6.5, Apache's Xerces-J 2.7.1, Apache's Xalan-J 2.7.0, and Apache's FOP 0.20.5, along with JAI 1.1.2 and JIMI (two image libraries used by FOP).[4,5]

---

[2] These commands assume that you already created (for Project 1) a directory called "`cscie259`" in your FAS account's home directory.

[3] Beware the distinction between `~cscie259` and `~/cscie259`. Also, if you do your work on a Windows machine, take care to substitute backslashes for any forward slashes as well as `dir` for any instances of `ls` in this document's directions.

[4] Specifically, if your local machine runs Linux or UNIX: define an environment variable called "`JAVA_HOME`" that points to your installation of J2SE SDK 1.4.2; create (in accordance with Sun's "Endorsed Standards Override Mechanism") in the SDK's `jre/lib/` directory a directory called `endorsed`, thereafter copying into it `serializer.jar` and `xalan.jar` from your installation of Xalan-J 2.7.0 as well as `xercesImpl.jar` and `xml-apis.jar` from your installation of Xerces-J 2.7.1; define an environment variable called "`ANT_HOME`" that points to your installation of Ant 1.6.5; define an environment variable called "`FOP_HOME`" that points to your installation of FOP 0.20.5; define an environment variable called "`JAI_HOME`" that points to your installation of JAI 1.1.2; define an environment variable called "`JIMI_HOME`" that points to your installation of JIMI; define an environment variable called "`JAVACMD`" whose value is "`$JAVA_HOME/bin/java`"; add "`.`", "`$FOP_HOME/build/fop.jar`", "`$FOP_HOME/lib/batik.jar`", "`$FOP_HOME/lib/avalon-framework-cvs-20020806.jar`", "`$JAI_HOME/lib/jai_core.jar`", "`$JAI_HOME/lib/jai_codec.jar`", and "`$JIMI_HOME/JimiProClasses.zip`" to your `CLASSPATH`; and prepend "`$JAVA_HOME/bin`" and "`$ANT_HOME/bin`" to your `PATH`.

[5] Specifically, if your local machine runs some version of Windows: be sure you've installed J2SE SDK 1.4.2 in the root of your hard drive (or, at least, not within any directory whose name contains spaces); create (in accordance with Sun's "Endorsed Standards Override Mechanism") in the SDK's `jre\lib\` directory a directory called "`endorsed`", thereafter copying into it `serializer.jar` and `xalan.jar` from your installation of Xalan-J 2.7.0 as well as `xercesImpl.jar` and `xml-apis.jar` from your installation of Xerces-J 2.7.1; define an environment variable called "`JAVA_HOME`" that points to your installation of J2SE SDK 1.4.2; define an environment variable called "`ANT_HOME`" that points to your installation of Ant 1.6.5; define an environment variable called "`FOP_HOME`" that points to your installation of FOP 0.20.5; define an environment variable called "`JAI_HOME`" that points to your installation of JAI 1.1.2; define an environment variable called "`JIMI_HOME`" that points to your installation of JIMI; define an environment variable called "`JAVACMD`" whose value is "`%JAVA_HOME%\bin\java`"; add "`.`", "`%FOP_HOME%\build\fop.jar`", "`%FOP_HOME%\lib\batik.jar`",

Notice, now, that your `project2-6.0/` directory is structured as follows.

```
project2-6.0/
  b2b/
  myblockbuster/
    gif/
    jpg/
    xhtml/
    xml/
    xsl/
  xtube/
    gif/
      logos/
      map/
      miscellany/
    pdf/
    svg/
    wav/
    xhtml/
    xml/
    xsl/
```

Ensure that your account or machine is in order by executing the following command from within any directory.[6]

```
EnvironmentCheck
```

Confirm that your account or machine is configured to use Xerces-J 2.7.1 and Xalan-J 2.7.0. Then, execute the following command from within any directory.[7]

```
xalan
```

You should then see the command-line options for Xalan-J's `Process` class.

If your account or system does not behave as described above, simply contact `cscie259@lists.dce.harvard.edu` or the staff for assistance.

"`%FOP_HOME%\lib\avalon-framework-cvs-20020806.jar`", "`%JAI_HOME%\lib\jai_core.jar`", "`%JAI_HOME%\lib\jai_codec.jar`", and "`%JIMI_HOME%\JimiProClasses.zip`" to your `CLASSPATH`; and prepend "`%JAVA_HOME%\bin`" and "`%ANT_HOME%\bin`" to your `PATH`.

[6] Note that, on `nice.fas.harvard.edu`, `` `EnvironmentCheck` `` is an alias for `` `java org.apache.xalan.xslt.EnvironmentCheck` ``.

[7] Note that, on `nice.fas.harvard.edu`, `` `xalan` `` is an alias for `` `java org.apache.xalan.xslt.Process` ``. If you opt to develop on a Windows machine, you can achieve this same aliasing effect by creating in any directory in your `PATH` a file called `xalan.bat` containing the following seven lines.

```
@echo off
:rep
shift
set xalanargs=%xalanargs% %0
if not !%1==! goto rep
java org.apache.xalan.xslt.Process %xalanargs%
set xalanargs=
```

Quickies.

Please place your answers to questions 3 and 4 in a file called `questions.xml` in your `project2-6.0/` directory. Then, in that same directory, create an XSLT stylesheet called `questions.xsl` that, given `questions.xml` as input, formats your answers as XSL-FO in a file called `questions.fo`. Finally, run `questions.fo` through FOP 0.20.5 in order to generate `questions.pdf` in the same directory.

In other words, after placing your answers in `questions.xml`, execute the following command.

```
xalan -IN questions.xml -XSL questions.xsl -OUT questions.fo
```

Then, convert your XSL-FO to PDF by executing the command below.[8]

```
fop -fo questions.fo -pdf questions.pdf
```

The format of your answers is entirely up to you. Anything neat, however simple, is acceptable.

Okay, questions 3 and 4 refer to the XML below. You are welcome to check your answers with any XPath processor, but we strongly recommend that you attempt to answer the questions on your own first.

```
<students>
    <student id="1">
        <name>
            <last>Bond</last>
            <first>James</first>
        </name>
        <GPA>3.8</GPA>
    </student>
    <student id="55">
        <name>
            <last>Wiggins</last>
            <first>Ender</first>
        </name>
        <GPA>3.8</GPA>
    </student>
</students>
```

[8] Note that, on `nice.fas.harvard.edu`, `` `fop` `` is an alias for `` `java org.apache.fop.apps.Fop` ``. Also, although we do require that you output `questions.fo`, know that, during development, you can avoid outputting `questions.fo` altogether by executing the command below, which will compel FOP to execute Xalan-J for you.

```
fop -xml questions.xml -xsl questions.xsl -pdf questions.pdf
```

3. (2 points.) Consider the XML above.

   i. Write an inefficient expression that selects all of the students (*i.e.*, `student` elements) whose last name starts with `'B'`.[9]

   ii. Write an efficient expression that selects all of the students (*i.e.*, `student` elements) with GPAs of 3.7 or higher whose last name starts with `'B'`.[10]

4. (7 points.) Consider again the XML above. But assume now that the context node is Ender's `name` element (the result of selecting `/students/student[@id='55']/name`).

   i. How many nodes are returned by selecting `"ancestor::*"`?

   ii. How many nodes are returned by selecting `"preceding::*"`?

   iii. Write any expression that evaluates to the fourth node returned by `"preceding::*"`.

   iv. How many nodes are returned by selecting `"following::*"`? How many nodes are returned by selecting `"following::node()"`? What is the fundamental difference between the two expressions?

   v. Re-write the following expression (which, unfortunately, wraps on to two lines) using abbreviated syntax.

```
/child::students/descendant-or-self::node()/self::name[child::last =
'Bond']/parent::node()/attribute::id
```

[9] By "inefficient," we mean an expression that performs unnecessary work; rest assured that many inefficient answers are possible.
[10] By "efficient," we mean an expression that doesn't perform unnecessary work.

It's Time for B2B!

5. (20 points.) Business-to-business (B2B) standards define XML schemas (*i.e.*, formats) for business documents. Unfortunately, many such standards exist. Companies participating in B2B exchanges, however, must agree on a document format before they can communicate. The solution for many companies is to have one, internal format for their documents and several, external formats for exchanges with partners. Every time a document (*e.g.*, a purchase order) is sent to a partner, a document in its internal format is translated to the required external format; every time a document is received, the reverse of this process must occur.

   Not surprisingly, most business documents include some specification of date and time (*e.g.*, an element specifying the document's creation time, a product's ship date, *etc.*). Unfortunately, companies' formats for dates and time differ all too often as well. For instance, Ariba Inc., a provider of online marketplaces that controls the cXML (commerce eXtensible Markup Language) protocol, utilizes the ISO 8601:2000 format for dates and times.[11] Meanwhile, the Open Application Group, a "non-profit consortium focusing on best practices and processes based on XML content for eBusiness and Application Integration," has defined its own XML-based representation of dates and times in its OAGIS (Open Applications Group Integration Specification) standard.[12]

   For instance, according to the ISO 8601:2000 format, the representation of 6:39:09 P.M. on 12 March 1999 in the time zone 8 hours behind UTC (Universal Coordinated Time) would be the following.

```
1999-03-12T18:39:09-08:00
```

   As the above suggests, ISO 8601:2000 specifies that a year, month, and day, respectively, are to be separated by `'-'`; that date and the time are to be separated by `'T'`; the time is to be represented as hours, minutes, and seconds, respectively, each separated by `':'`; the offset from UTC is to follow. (Note that the `'-'` preceding the time zone's offset effectively denotes west, whereas a `'+'` would denote east.)

   Meanwhile, according to OAGIS, the representation of this same date and time would resemble the following.

```
<DATETIME qualifier="SHIP">
    <YEAR>1999</YEAR>
    <MONTH>03</MONTH>
    <DAY>12</DAY>
    <HOUR>18</HOUR>
    <MINUTE>39</MINUTE>
    <SECOND>09</SECOND>
    <TIMEZONE>-0800</TIMEZONE>
</DATETIME>
```

[11] See `http://www.ariba.com/` and `http://www.cxml.org/` for more information, if interested.
[12] See `http://www.openapplications.org/` for more information, if interested.

Suppose, now, that your company utilizes OAGIS's `DATETIME` format and, of course, you've just partnered with a firm that uses ISO 8601:2000. Write an XSLT stylesheet that takes as its sole parameter a date and time in ISO 8601:2000 format (simplified as in our example) and converts it to a `DATETIME` element (simplified as in our example, with the value of the element's `qualifier` attribute hard-coded as `"SHIP"`). To be clear, it is not necessary to research ISO 8601:2000 or OAGIS's `DATETIME` format; assume the simplified formats described herein.

Notice that, in `project2-6.0/b2b/b2b.xsl`, we've provided you with a framework for this converter. To test your code, simply execute a command like the below from within your `project2-6.0/b2b/` directory.

```
xalan -XSL b2b.xsl -PARAM datetime "1999-03-12T18:39:09-08:00"
```

The results will be printed to `STDOUT`. (They, of course, should be pretty-printed.)

My Blockbuster.

6. (30 points.) In Lecture 4, we saw the limitations of CSS with respect to the presentation of My Blockbuster's XML database. But we also glimpsed XSLT's far greater potential for this same task.

   Unfortunately, you didn't do such a good job converting dates and times, so the employer of question 5 is now your former employer. So, of course, you took a job at My Blockbuster. Your first assignment is to get My Blockbuster ready for the World Wide Web. But your boss has asked that you display the company's movies in a number of different formats in one long webpage. Specifically, your boss wants visitors to be able to view My Blockbuster's holdings alphabetically by title, alphabetically by actor, or categorized by rating. At the top of the webpage should be hyperlinks to each of these three views.[13]

   Moreover, your boss wants visitors, when viewing by title, to see not only a movie's name, but also its lead actor(s) and rating, as well as at least three of the following: genre, rating, summary, details, year, director, studio, user rating, length, format, price, and/or quantity in stock. Plus, he wants each actor's name hyperlinked to his or her name in your page's view by actor, so that visitors can find an actor's other movies with a single click; and he wants each rating hyperlinked to the category of other movies with that rating.

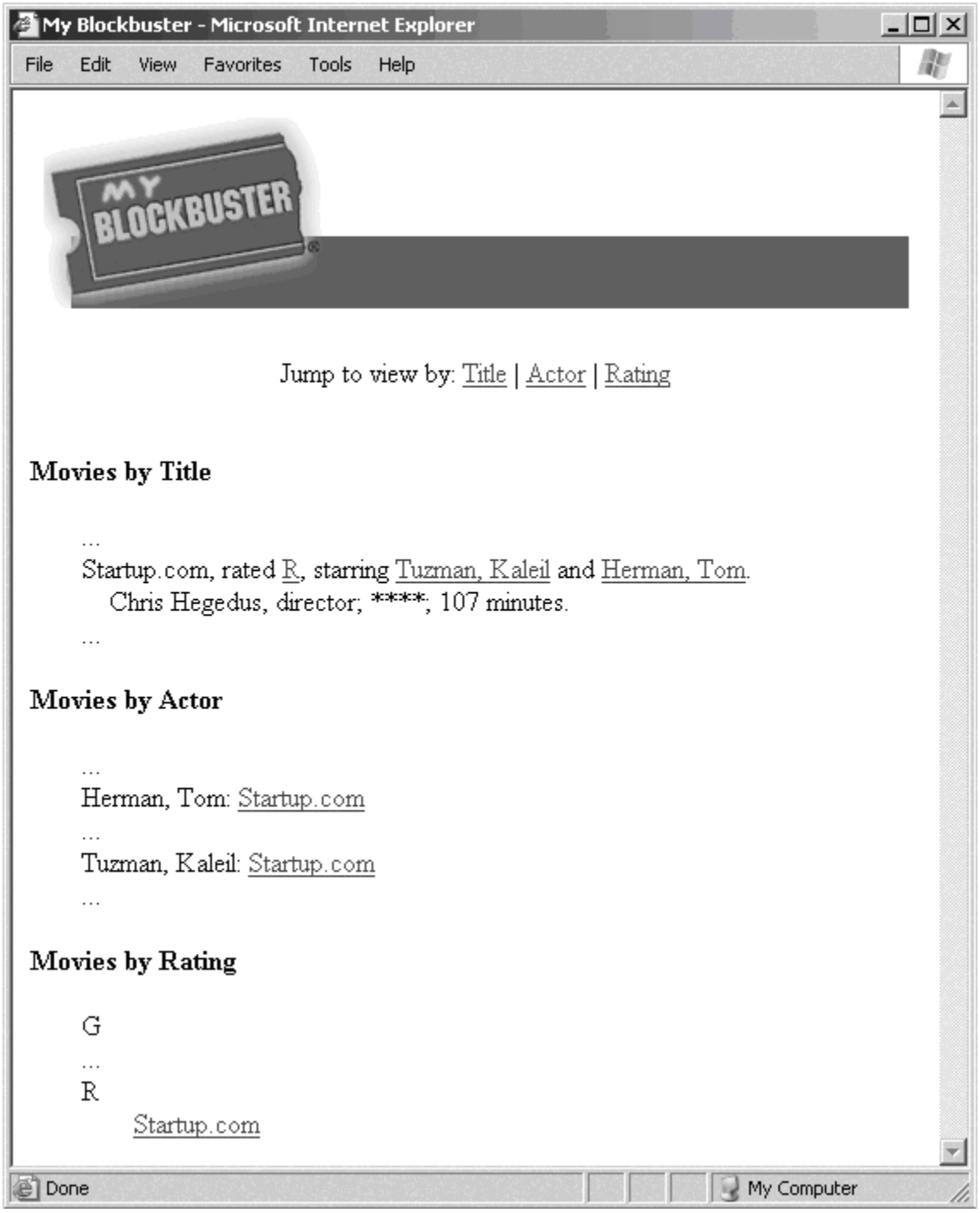
   As for the view by actor, your boss is fine with your simply displaying each actor's name, followed by a list of that actor's movies (each of which, of course, is hyperlinked to that movie's listing in your view by title).

   Similarly, with respect to your view by rating, your boss is fine with your simply displaying each rating, followed by a list of movies with that rating (each of which, again, is hyperlinked to that movie's listing in your view by title).

   To be clear, your boss wants a webpage resembling the below.[14]

[13] In other words, the top of your page should contain links to fragment IDs like `<a href="#title">Title</a>`, `<a href="#actor">Actor</a>`, and `<a href="#rating">Rating</a>`; lower in the page should be anchors like `<a name="title" />`, `<a name="actor" />`, and `<a name="rating" />`.

[14] Works of greater beauty, while not required, are encouraged.



Rather than create a nightmare for yourself by hand-coding this webpage, you've decided to take your newfound skills with XSLT out for a spin, so that displaying new releases simply requires updates to the company's XML database.

Notice that, in `project2-6.0/myblockbuster/`, we've provided you with `gif/myblockbuster.gif`, `xml/myblockbuster.xml`, and `xsl/myblockbuster.xsl`, the

last of which already contains a framework for this assignment. While you are free to modify `myblockbuster.xsl` (and, if you're a better artist than we, `myblockbuster.gif`), you may not modify `myblockbuster.xml`. As the framework suggests, your XSLT stylesheet's output should be XHTML 1.0. However, you are free to use any tag supported by the latest version of version of Firefox, Internet Explorer, Mozilla, Netscape Navigator, Opera, or Safari; we will not validate your XSLT stylesheet's output against XHTML 1.0's DTD.[15] Needless to say, your XSLT stylesheet and its output should be commented and pretty-printed.[16]

To apply your XSLT stylesheet to My Blockbuster's XML database (thereby testing the former), simply execute the following from within `project2-6.0/`.

```
ant transform-myblockbuster
```

The results will be saved in `project2-6.0/myblockbuster/xhtml/myblockbuster.html`.

Take a look at `project2-6.0/build.xml` to see exactly what Ant is (and can be) doing for you.

Brownie points (and possibly brownies) may be awarded for submissions that make aesthetically pleasing use of the JPEGs in `project2-6.0/myblockbuster/jpg/`, whose file names happen to appear in `image` elements in `myblockbuster.xml`.[17]

[15] You, however, may, if you are so inclined; simply visit `http://validator.w3.org/`.
[16] Notice our inclusion of `xmlns:xalan="http://xml.apache.org/xslt"` as an attribute of `xsl:stylesheet` and of `xalan:indent-amount="4"` as an attribute of `xsl:output` in `myblockbuster.xsl`. Xalan-J's default level of indentation, unfortunately, is 0; without these attributes, then, your output would not be pretty-printed even with `xsl:output`'s `indent` attribute set to `"yes"`. See `http://xml.apache.org/xalan-j/usagepatterns.html#outputprops` for more information.
[17] Images from `http://www.imdb.com/` and `http://www.amazon.com/`.

"Mind the gap."

7. (0 points.) If you can believe it, you actually did a worse job at My Blockbuster than you did converting dates and times. In fact, you haven't only been fired this time. You've been deported![18]

   Fortunately, you scored a job with the London Underground (a.k.a. the Tube). The Tube is a complex web of interconnecting stations and lines running beneath London. In fact, take a look for yourself by checking out, in `project2-6.0/xtube/pdf/`, `large_print_map03.pdf` (which highlights the Tube and Docklands Light Railway), `connections.pdf` (which highlights Tramlink and National Rail), and `london_connections.pdf` (which depicts everything).

   As luck would have it, your first (and probably last) assignment is to make sense of the Tube's interconnections. Specifically, you've been tasked with putting together XSLT-generated SVG and XHTML that detail greater London's tubes, trains, and trams (within the confines of zones 1 through 6).

   Included, then, in `project2-6.0/xtube/xml/` is `xtube.xml`, a database replete with details on greater London's lines and stations. Atop this file can be found each line's unique identifier, name, type, logo, and URL. Also noted for each Underground line is its official color. The excerpt below elucidates.

```
<london>
  <lines>
    <line>
      <lineid>1</lineid>
      <name>Bakerloo</name>
      <color>#A57536</color>
      <type>Tube</type>
      <logo type="GIF">lulogo.gif</logo>
      <url>http://tube.tfl.gov.uk/</url>
    </line>
    [...]
  </lines>
  [...]
</london>
```

   Needless to say, this excerpt is for Bakerloo, an Underground line whose unique identifier is 1, whose color is `#A57536`, whose logo is `lulogo.gif`, and whose URL is `http://tube.tfl.gov.uk/`.

   Also in `xtube.xml` can be found each station's unique identifier, name, coordinates, and neighbors. Note that some stations belong to more than one line and that stations can have one or more neighbors. The excerpt below elucidates.

[18] Okay, this joke has run its course now.

```
<london>
  [...]
  <stations>
    [...]
    <station>
      <stationid>100</stationid>
      <stationname>Sloane Square</stationname>
      <coordinates>
        <os>
          <easting>528081</easting>
          <northing>178666</northing>
        </os>
        <gif>
          <x>984</x>
          <y>970</y>
        </gif>
      </coordinates>
      <neighbor>
        <stationid>99</stationid>
        <lineid>3</lineid>
      </neighbor>
      <neighbor>
        <stationid>99</stationid>
        <lineid>4</lineid>
      </neighbor>
      <neighbor>
        <stationid>17</stationid>
        <lineid>3</lineid>
      </neighbor>
      <neighbor>
        <stationid>17</stationid>
        <lineid>4</lineid>
      </neighbor>
    </station>
    [...]
  </stations>
</london>
```

Needless to say, this excerpt is for Sloane Square, a station on the Underground's Circle Line (whose unique identifier is 3) and District Line (whose unique identifier is 4). Sloane Square's neighboring stations, via either line, are South Kensington (whose unique identifier is 99) and Victoria (whose unique identifier is 17). The coordinates of Sloane Square on Great Britain's Ordnance Survey (OS) National Grid, meanwhile, are (E,N) = (528081,178666), where E denotes eastings and N denotes northings. The coordinates of Sloane Square on `project2-6.0/xtube/gif/map/big.gif`, on the other hand, are (x,y) = (984,970), where (0,0) denotes that image's top-left-hand corner.

Not familiar with Great Britain's grid? Well, this grid partitions Great Britain into ninety-one squares with sides of 100 kilometers (km), as per Figure 8 from *A guide to coordinate systems in Great Britain*, reprinted below.[19]

[19] See `http://www.gps.gov.uk/additionalInfo/images/A_guide_to_coord.pdf`.

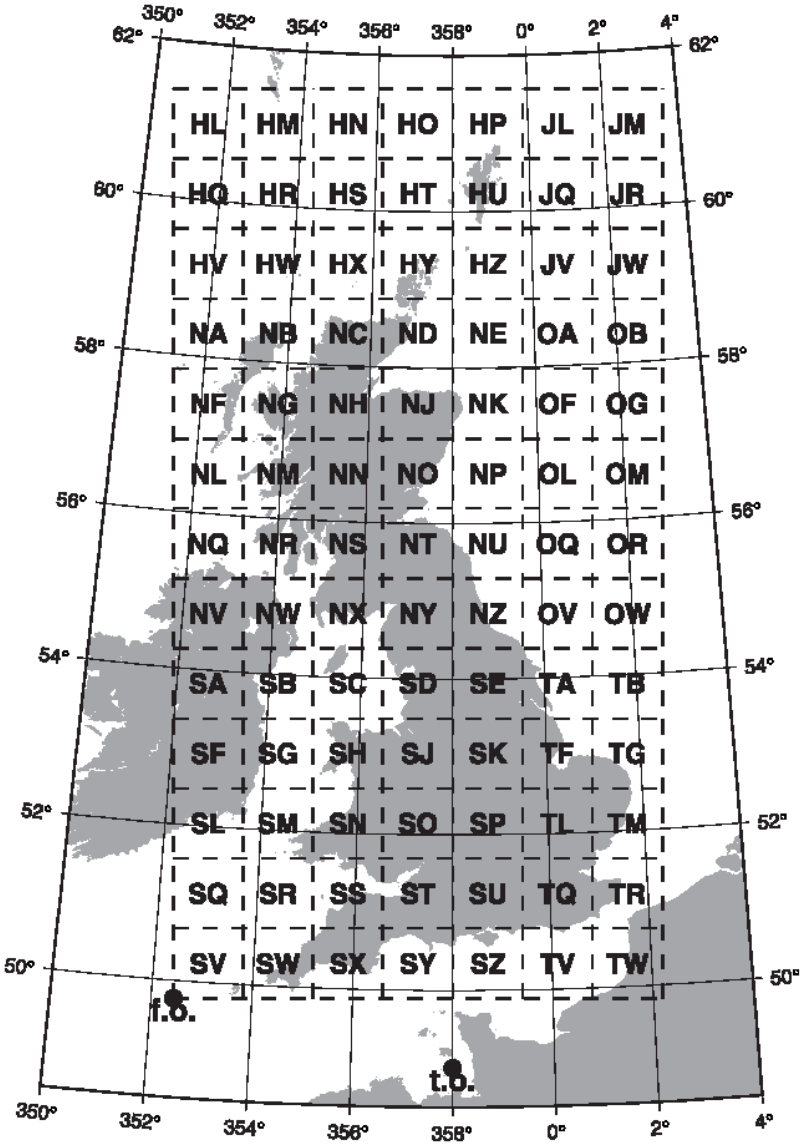
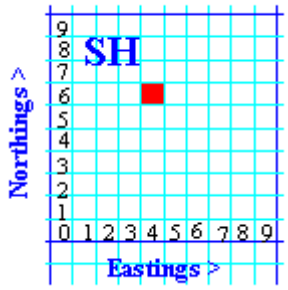


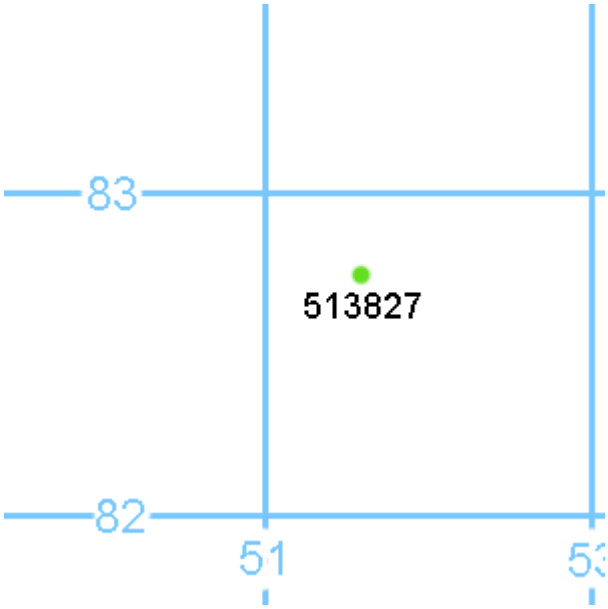
Figure 8: The National Grid, showing the true origin (t.o.) and false origin (f.o.)

Each of those squares is then divided into "sheets" with sides of 10 km. For instance, the figure below depicts (*E*,*N*) = (24,36), where the leading 2 and leading 3 in these coordinates derive from the corresponding sheet, which, in this case, is SH.[20]

[20] Image from `http://www.aditsite.co.uk/html/osgrid.html`.



Additional granularity is possible, however, as each of these squares with sides of 10 km can be divided into squares with sides of 1 km. For instance, the figure below depicts (*E*,*N*) = (513,827), which can also be written as 513827.[21]



Of course, squares with sides of 10 km can, in turn, be further divided, eventually down to squares with sides of 1 meter (m). For instance, the figure below depicts (*E*,*N*) = (528081,178666), the coordinates of Sloane Square (highlighted therein with an arrow).[22] (On a Landranger map, these coordinates would be written as TQ30248299, inasmuch as Sloane Square is in sheet TQ of the grid.)

[21] Image from `http://website.lineone.net/~galaxypix/maprefs.html`.
[22] Image from `http://www.streetmap.co.uk/newmap.srf?x=528081&y=178666`.

Alright, now that you (hopefully) understand the data with which you've been provided, it's time to put it to use.

Your British boss has demanded that you enable him to transform the data into SVG or XHTML, depending on his mood. In `project2-6.0/xtube/xsl/`, then, is `xtube.xsl`, a framework for meeting these demands. Notice that this stylesheet is designed to produce either SVG or XHTML, depending on the value of its `mode` parameter.

Unfortunately, at present, it produces little of either. Confirm as much by executing the following from within `project2-6.0/`.

```
ant transform-xtube-xhtml
```

You should see only that a framework for XHTML has been outputted to `project2-6.0/xtube/xhtml/xtube.html`.

Next execute the following, also from within `project2-6.0/`.

```
ant transform-xtube-svg
```

You should see only that a framework for SVG has been outputted to `project2-6.0/xtube/svg/xtube.svg`.

Take a look at `project2-6.0/build.xml` to see exactly what Ant is (and can be) doing for you. And take a look in `project2-6.0/xtube/wav/` for a few Easter eggs.[23]

8. (35 points.) To appease your boss, proceed to augment `xtube.xsl` with XSLT so that, when parameterized with a `mode` of `xhtml`, the stylesheet outputs a webpage detailing London's tubes, trains, and trams. Specifically, this webpage must list, for each line, the line's stations. This webpage must also list, for each station on a line, the station's neighbors on that line. (Inasmuch as lines can fork or cycle, you'll find that an end-to-end representation of London's lines is difficult: it suffices, then, to list each line's stations alphabetically.) Moreover, if it is

[23] Sounds from `http://www.geocities.com/CollegePark/3812/tubesounds.html`.

possible to transfer at some station from one line to another, this webpage should offer a hyperlink (*i.e.*, a fragment ID) from the former's list of stations to the latter's. Of course, each line's colour, if any, should somehow be incorporated into the page, as should each line's logo.

And, so that your boss might find his way whilst traveling in London, using your webpage for direction, visitors should also be able to follow for each station a hyperlink (based on that station's eastings and northings) to its StreetMap on `www.streetmap.co.uk`. For instance, a StreetMap for Sloane Square, whose coordinates are (*E*,*N*) = (528081,178666), can be found at the URL below.

```
http://www.streetmap.co.uk/newmap.srf?x=528081&y=178666
```

For additional options for linking to `www.streetmap.co.uk`, see the URL below.

```
http://www.streetmap.co.uk/linkto.htm#gridreference
```

Of course, of interest to visitors as well is each station's location on the Underground's official map, provided as `project2-6.0/xtube/gif/map/big.gif`. Most people, however, don't enjoy looking at GIFs that are 2800 pixels by 2000 pixels. Also included in `project2-6.0/xtube/gif/map/`, then, are pieces of that map, each roughly 400 pixels by 400 pixels. (Some are a bit bigger so that stations' names aren't split across pieces.) Know that `1_1.gif` represents the top-left-hand corner of `big.gif`, `1_5.gif` represents the bottom-left-hand corner of `big.gif`, and whereas `7_5.gif` represents the bottom-right-hand corner of `big.gif`.

A station at (*x*,*y*), then, is best viewed in the GIF with filename `m_n.gif`, where `m` = $\lfloor x/400 \rfloor + 1$ and `n` = $\lfloor y/400 \rfloor + 1$. Accordingly, Sloane Square, whose coordinates are (*x*,*y*) = (984,970), is best viewed in `3_3.gif`, since $\lfloor 984/400 \rfloor + 1 = 3$ and $\lfloor 970/400 \rfloor + 1 = 3$.

Be sure, then, that visitors can follow for each station a hyperlink to the appropriate piece of `big.gif`; that hyperlink may be an absolute reference to the GIF itself or a fragment ID, if you opt to incorporate the pieces into your webpage.

Needless to say, your British boss hopes that your XHTML will be valid, according to the W3C's DTD for XHTML 1.0 Transitional, but he'll understand if you opt to employ tags or attributes that aren't in the W3C's Recommendation. He does expect, though, that your stylesheet and its output be commented and pretty-printed.

If `big.gif` and its many pieces leave you puzzled as to some aspect of London's tubes, trains, and trams, you may find the searchable PDFs in `project2-6.0/xtube/pdf/` helpful.

Moreover, you can find maps for individual lines via most of the URLs within `xtube.xml`.

You are welcome to add files to `project2-6.0/xtube/`, particularly additional maps and images. But you may not modify `xtube.xml`.

That you have not been provided with more precise specification of your page's appearance and functionality is intentional. Creative interpretations are encouraged. Although things of beauty are encouraged, your work will not be evaluated with respect to aesthetics but with respect to the quality of its underlying design and the extent to which it fulfills your British boss's demands. Impress us. Er, him.

9. (35 points.) How silly of you to have been deferring to others' maps when you have all that you need to generate those maps yourself!

   Your boss's final demand, then, is that you augment `xtube.xsl` with XSLT so that, when parameterized with a `mode` of `svg`, the stylesheet outputs a map in the form of SVG detailing the actual geography of London's tubes, trains, and trams. Specifically, this SVG, when viewed with Adobe SVG Viewer 3.0x, should make clear each line's route and interconnections, as well as each station's name and geographically accurate location (relative to other stations), in the spirit of Lecture 6's depiction of the Circle line[24] That is, your map should utilize each station's OS coordinates for positioning and not those for `big.gif`. Of course, each line's colour, if any, should somehow be incorporated into your SVG.

   For guidance with your design, recall Lecture 6's representation of the Circle line's route in SVG, albeit sans interconnections.

   You need not worry if lines or stations' names overlap, but efforts toward minimizing overlap are encouraged.

   May we also suggest, but not require, that your map include a legend for lines as well as hyperlinks for each station to a StreetMap or pieces of `big.gif`. In short, allow us to suggest, but not require, that you make your SVG as interesting and interactive as possible!

   We leave it to you to determine how to translate each station's OS coordinates into values more useful. In other words, outputting SVG with height and width in the hundreds of thousands is not acceptable.

   At some point, you will realize that your map doesn't exactly look like London's official maps for its tubes, trains, and trams. Do not despair! Recall from Lecture 6 that official maps (including `big.gif` and the PDFs in `project2-6.0/xtube/pdf/`) distort the city's geography in order to present lines and stations more clearly. In fact, surf on over the URL below for a bit of history (and morphing)!

   `http://www.tfl.gov.uk/tube/maps/realunderground/realunderground.html`

   By the way, if you'd like to rasterize your SVG as a JPEG, PDF, PNG, or TIFF, know that you can use Apache's Batik 1.6, which is available for use on `nice.fas.harvard.edu` and

[24] If you opt to develop this project on a platform not supported by Adobe SVG Viewer 3.0x, be sure to test your SVG on a platform that does support Adobe SVG Viewer 3.0x prior to its submission. Although other viewers exist, as per `http://www.w3.org/Graphics/SVG/SVG-Implementations.htm8`, your SVG will only be tested with Adobe's viewer.

available for download via the course's website. For instructions on Batik's usage, browse the documentation available on the course's website or execute

```
batik-rasterizer
```

on `nice.fas.harvard.edu`.[25]

Of course, your boss is interested in your SVG, not its rasterization. So he doesn't expect you to make generate JPEGs, PDFs, PNGs, or TIFFs from your SVG. But he knows it's kind of fun.

10. (10 points.) In tackling questions 8 and 9, you likely discovered (bemoaned?) inefficiencies inherent in `xtube.xml` or, at least, structural aspects that rendered processing of `xtube.xml` by XSLT all the more challenging. Knowing what you know now, implement, in `converter.xsl`, a stylesheet that, when applied to `xtube.xml`, outputs a "better" version of the XML therein. By "better," we mean XML output that, had you been allowed to create and use it before now, would have facilitated implementation of `xtube.xsl`. Your conversion must not be lossy: it is unacceptable to lose any of the information implicit or explicit in `xtube.xml`.

    Be sure, in profuse comments laced throughout `converter.xsl`, to justify your design decisions. For instance, if you opt to restructure the `station` element, explain why; if your `converter.xsl`'s output contains redundant information, explain why; and so forth.

    To be clear, `xtube.xsl` must be designed to process the version of `xtube.xml` provided in this project's distribution, as per questions 8 and 9. You may not (re-)implement `xtube.xsl` to support your `converter.xsl`'s output.

    To apply your `converter.xsl` to `xtube.xml`, execute the following from within `project2-6.0/`.

    ```
    ant convert-xtube
    ```

    So that you can view it, the output will be saved in `xtube/xml/converted.xml`.

[25] Note that this command is just an alias for the command below, where `BATIK_HOME` points to Batik's installation.

```
java -Djava.awt.headless=true -jar $BATIK_HOME/batik-rasterizer.jar
```

Windows users with Batik installed on their local clients should instead execute

```
java -Djava.awt.headless=true -jar %BATIK_HOME%\batik-rasterizer.jar
```

after defining `BATIK_HOME`.

Submitting Project 2.

11. (0 points.) If you have not done your work on `nice.fas.harvard.edu`, transfer via SFTP your `project2-6.0/` directory from your local machine to the `cscie259/` directory in your FAS account's home directory, taking care to upload any text files in ASCII mode.

    Next, ensure that the structure of your `project2-6.0/` directory on `nice.fas.harvard.edu` is the following.

```
project2-6.0/
  b2b/
  myblockbuster/
    gif/
    jpg/
    xhtml/
    xml/
    xsl/
  xtube/
    gif/
      logos/
      map/
      miscellany/
    pdf/
    svg/
    wav/
    xhtml/
    xml/
    xsl/
```

    Your `project2-6.0/` directory should contain, at least, `build.xml`, `questions.fo`, `questions.pdf`, `questions.xml`, and `questions.xsl`; `project2-6.0/b2b/` should contain, at least, `b2b.xsl`; `project2-6.0/myblockbuster/` should contain, at least, all that it originally contained; and `project2-6.0/xtube/` should contain, at least, all that it originally contained.

    Particularly if you developed on another machine, ensure that all of your code transforms properly on `nice.fas.harvard.edu` by executing the appropriate commands.

    Finally, submit your work electronically by executing the following command from within your `project2-6.0/` directory.

```
cscie259submit project2
```

    Thereafter, follow any on-screen instructions until you receive visual confirmation of your project's successful submission. You may re-submit as many times as you'd like; each re-submission will overwrite any previous submission. But take care not to re-submit after the project's deadline, as only your latest submission's timestamp is retained.