ICT-4540 Homework 5

Purpose

This exercise will allow you to explore XQuery as a mechanism to extract information from an XML document (or multiple XML documents), and create useful applications of the data

What to Hand In

For each problem, please hand in your XQuery file, the resulting output of running the query against the data source, and the display of your data in a browser.
Your resulting documents should be well-formed, and you should check them to be sure.

Problems

Narrandera, New South Wales, Australia, Google Earth image

For the required exercise, we are using a public-domain data set from Australia, the ToiletmapExport. The data can be found at http://data.gov.au/, and a copy of it has been placed in our course materials for reference. It is quite a large data set, and regenerated on a regular basis. This is a typical XML document for processing.

All the other exercises are optional.

I. Simple extraction
   1. Create a file called *toilet.1.xquery* to contain an XQuery to extract all the toilets in the Australian town of Narrandera, a picturesque place near the River Red Gum National Park, along the Murrumbidgee River.
   2. Run *XQuery* against the ToiletmapExport.xml dataset
   3. The result set should be in the root element <*Toilets*>.
      Each entry will be in an element <*Toilet*> under the <*Toilets*> element.
      You will need to handle namespaces from the Australian document.
      You will want to ensure that the result document has nicely formatted output.
      The *Notes* section adds some details to these instructions.
   4. Extract the following fields in the following order into each <*Toilet*> element: The attributes Latitude and Longitude are extracted into separate elements under this element (as an aside, in case you wanted them to remain attributes, attributes must be extracted first to be successfully inserted in XQuery), the Town, the State, the Country (which is always Australia), the Name, and the FacilityType
   5. Ensure your output is a well-formed document.

II. Optional: More complex extraction
   1. The only thing that changes in this exercise from the previous one is the FLWOR expression.
   2. Instead of using the city of Narrandera as the anchor, instead, use its Latitude/Longitude to build a box 0.25 degree (about 15 miles) each direction from its center:
   3. Add a condition that either there must be a male and female accessible restroom, or an accessible unisex restroom.

4. Proceed as above.

III. Optional: Geographic context
   1. For either (or both) of the problem I or II solutions, make the output be valid KML
   2. Display your result file on Google Earth as your output.

IV. Optional: Multiple file manipulation
   1. This problem requires a copy of the *USPresidents.xml* and *PresidentBirthdays.xml* files in their original state.
   2. Goal: Create an XQuery to create the merged result document, where each President's birthday is inserted into the XML document in the correct location. The rest of these directions are a step-by-step recipe to accomplish this, if needed.
   3. Insert an XQuery comment for your name, class, date, and exercise number
   4. Set up copy-namespaces to no-preserve and inherit. When the problem is complete, you may want to see the result if you set to preserve.
   5. Set up the top-level element, presidents, and its end element, with a set of curly braces bracketing the content
   6. Set up two "let" statements; let $pres stand for the USPresidents.xml document, and $bd stand for the PresidentBirthdays.xml document.
   7. Set up a for loop letting $p stand for each of the president elements (presidents/president) in $pres
   8. Set up a let statement so $name will stand for the data content of the president's name
   9. Set up a let statement so $b will stand for the birthday element of the president such that the name equals $n
   10. Return the element president, with each of its subelements in a separate, curly-brace-enclosed statement (e.g., { $p/name }), inserting the birthday element, { $b }, in the right place
   11. You are ready to run the XQuery!
   12. Compare this result to what you did earlier. What are the anomalies? Which will be easier to maintain? Which was more work?
   13. Please submit your XQuery, the result, and answers to the questions in the previous step.

Notes

- You will find USPresidents.xml, PresidentBirthdays.xml, and president.css in the HWstarters ZIP file.
- You will find a snapshot of the Australian Government's ToiletmapExport.zip in the HWstarters ZIP file. Extract the contents, ToiletmapExport.xml, to a conducive place for performing the transformation.
- Running XQuery can be done in a number of tools. Oxygen is a very powerful one. XQDT is another one with a following. Among commercial tools, Mark Logic and Stylus Studio have adherents.
- This instruction will be for use of Saxon Home Edition (the free, open source one). The instructions are intended to be simple, but require you to download software, have a Java distribution on your computer, and require use of the Command window or Terminal application.
- Using Saxon
  - Download Saxon from saxon.sourceforge.net. These instructions assume you have downloaded the Java version (though the .Net version may work fine as well). A link for downloading the SaxonHE version (as a zip file) is at the top of the page.
  - Open a command prompt or terminal window
  - At the prompt, type *java -version* to ensure you have Java on your PATH. If you don't, please download and install Java to your machine. You will need only the JRE for this exercise.

- Unzip the Saxon zip file to a known location. We will use the saxon9he.jar file.
- If you are in the same directory as your XML file *ToiletmapExport.xml*, your XQuery *toilet.1.xquery* is in the same directory, and your *saxon9he.jar* file, the following command (all on one line) will run your XQuery:

```
java -cp saxon9he.jar net.sf.saxon.Query

 -q:toilet.1.xquery '!indent=yes'
```

  For Windows command line, please omit the single quotes (under Linux or Unix or MacOS they are required)
- Output will show up on the screen and can be copied to your result document (or redirected to a file)
- The ToiletmapExport.xml file uses namespaces heavily. Your XQuery must accommodate this. The first line of your XQuery should be:
  declare default element namespace "http://toiletmap.gov.au/" ; or *declare namespace t = "http://toiletmap.gov.au/" ;*. The second form will require you to use *t:* to qualify document elements in your XPaths.
- If you run your XQuery, you may find that the output puts namespace declarations on every node. You can avoid this by adding them to the top level node:
  <Toilets xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema-datatypes"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
- If you want to explore XQuery capabilities to preserve and inherit namespaces, look up the XQuery declaration *declare copy-namespaces*.

- KML Specification can be found at: https://developers.google.com/kml/documentation or www.opengeospatial.org/standards/kml/
- For the USPresident and PresidentBirthdays document integration, you will note some things that don't work quite as expected for the output document. Do you see why these occurred? What is the right fix? What could have been done to avoid this?

Evaluation

Criteria	Weight
Simple Extraction: XQuery instructions	20%
Simple Extraction: root element and namespaces	30%
Simple Extraction: elements	30%
Simple Extraction: well-formed output document	20%
More complex extraction	+10%
Geographic context	+10%
Multiple File Integration	+10%