CompSci 316 Fall 2016: Homework #3

100 points (8.75% of course grade) + 20 points extra credit
Assigned: Thursday, October 20
Due: Tuesday, November 8

This homework should be done in parts as soon as relevant topics are covered in lectures. If you wait until the last minute, you might be overwhelmed.

For Problem 1, you will need to use Gradiance. Access Gradiance via the "Gradiance" link on the course website. There is no need to turn in anything else for these problems; your scores will be tracked automatically.

For other problems, you will need to turn in the required files electronically. Please read the "Help → Submitting Non-Gradiance Work" section of the course website for instructions. When submitting your work, make sure you select the correct course and homework. Multiple submissions are okay, but please upload ***all*** required files in each resubmission.

Problems 2, 3, X1, and X2 should be completed on your course VM. Before you start, make sure you refresh your VM, by logging into your VM and issuing the following command:

```
/opt/dbcourse/sync.sh
```

Problem 1 (15 points)

Complete the Gradiance homework titled "Homework 3.1 (XML)."

Problem 2 (55 points)

In `/opt/dbcourse/assignments/hw3/` on your VM, you will find an XML file `congress.xml` containing information about the current (114th) US Congress. Logically, the file consists of two sections:

- Each `person` element under `congress/people` stores information about a legislator, including the roles he or she has served in the Congress. A `role` with type "`rep`" indicates a Representative (member of the House), while a `role` with type "`sen`" indicates a Senator (member of the Senate). A `role` is current if its `current` attribute equals 1.
- Each `committee` element under `congress/committees` stores information about a committee. It has a list of members, whose ids reference those of `person` elements in the first section; `role` specifies the role of the member in the committee (e.g., chair or ranking member). Oftentimes a committee can be divided into subcommittees. Each `subcommittee` element has its own list of members, which should be a subset of the committee members. A legislator can serve on multiple committees, and even multiple subcommittees under the same committee.
    - Note that there are some "dangling" `person` references from under `congress/committees`. Representative Mark Takai passed away in May 2016; Representative Ed Whitfield resigned in September 2016. But they still remained on some committee rosters.

Please refer to the document "XML Tips" on the course Web site for instructions on running `saxonb-xquery`, the Saxon XQuery processor. Write queries in XQuery to answer the following questions. Because Saxon does not use any indexes and does not have a sophisticated optimizer, query performance may be heavily influenced by the way you write your queries. If a particular query takes forever to run, consider reordering loops and evaluating selections (filters) as early as possible. Note that you can add comments to your queries by enclosing them in "`(:`" and "`:)`".

For each question below, say (a), write your XQuery in a file named `2a.xq`, and generate the output file `2a.xml` by running

```
saxonb-xquery -s /opt/dbcourse/assignments/hw3/congress.xml 2a.xq > 2a.xml
```

Turn in all your `.xq` and output `.xml` files.

(a) Find all legislators whose name ends with "`Price`". (For each of them, simply print the entire `person` element.) You can use `ends-with(str1, str2)` to test if `str1` ends with `str2`.

(b) Find the legislator who serves as the role of "`Chairman`" on the Senate Select Committee on Intelligence (code name "`SLIN`"). Simply print the entire `person` element.

(c) List all current female Senators born before 1940. Format each of them as an element of the form `<senator name="…"/>`. You can use `xs:date("1939-12-31") < xs:date("1940-01-01")` to test if the date 1939-12-31 precedes the date 1940-01-01.

(d) List the name, district, and party of each current Representative of NC. Format each of them as an element of the form `<representative name="…" district="…" party="…"/>` and sort them according to the district.

(e) List the names of current Senators who at some point also served as Representatives. Format each of them as an element of the form `<member>…</member>`.

(f) Find the number of current Representatives for each party. For each party, format the output as an element of the form `<record count="…" party="…"/>`.

(g) List the names of legislators who are NOT serving in any committee or subcommittee. Format each of them as an element for the form `<person>…</person>`.

Problem 3 (30 points)

Continuing from the last problem, your job is to produce an output XML file `percom.xml`, which presents information about legislators and their committee assignments in a more concise and readable form. The output file should be structured as follows, and conform to the DTD in `/opt/dbcourse/assignments/hw3/percom.dtd`.

- The root element is `congress`.
- `congress` has two child elements: `house` and `senate`, each listing its current legislators. See the description of `congress.xml` above for how to determine who are current members of the two chambers.
- Each legislator is represented as a `person` element, with a name attribute whose value is taken from `person/@name` in `congress.xml`. Under `person`, list each committee that this legislator serves in as a `committee` element. A `committee` element has a `name` attribute whose value is taken from `committee/@displayname` in `congress.xml`; it also has a `role` attribute whose value is taken from `member/@role` (or simply "`Member`" if no role is specified). Under `committee`, list each

subcommittee of the committee that this legislator serves in, as a `subcommittee` element. Like a `committee` element, a `subcommittee` has a `name` attribute and a `role` attribute.

For example, here is a snippet of the output showing the committee assignment for Bernie:

```
<?xml version="1.0" encoding="UTF-8"?>
<congress>
   <house>
      …
   </house>
   <senate>
      …
      <person name="Bernard Sanders">
         <committee name="Senate Committee on the Budget" role="Ranking Member"/>
         <committee name="Senate Committee on Energy and Natural Resources" role="Member">
            <subcommittee name="Water and Power" role="Member"/>
            <subcommittee name="National Parks" role="Member"/>
            <subcommittee name="Energy" role="Member"/>
         </committee>
         <committee name="Senate Committee on Environment and Public Works" role="Member">
            <subcommittee name="Clean Air and Nuclear Safety" role="Member"/>
            <subcommittee name="Transportation and Infrastructure" role="Member"/>
            <subcommittee name="Fisheries, Water, and Wildlife" role="Member"/>
         </committee>
         <committee name="Senate Committee on Health, Education, Labor, and Pensions" role="Member">
            <subcommittee name="Primary Health and Retirement Security" role="Ranking Member"/>
            <subcommittee name="Children and Families" role="Member"/>
         </committee>
         <committee name="Senate Committee on Veterans' Affairs" role="Member"/>
      </person>
      …
   </senate>
</congress>
```

To generate `percom.xml` from `congress.xml`, you have the following options:

(a) Write a Python program using SAX API (`xml.sax`).
(b) Write a Python program using DOM API (`xml.dom`).
(c) Write an XQuery.
(d) Write an XSLT program.

Your code should handle any potential dangling references mentioned in the last problem. Please refer to the document "XML Tips" on the course website for instructions on how to write and run these programs and queries. You should validate your output file `percom.xml` against the provided `percom.dtd`, using the following command (more information on `xmllint` can be found in "XML Tips")

```
xmllint --dtdvalid /opt/dbcourse/assignments/hw3/percom.dtd --noout percom.xml
```

You must implement two out of the four options. For each option you implement, submit source code and output.

Extra Credit Problem X1 (10 points)

Implement the other two options that you left out for Problem 3.

Extra Credit Problem X2 (10 points)

As part of the trendy "NoSQL" movement, CompSci 316 decides to port the beer drinker database to XML. For reference, here is the relational schema:

Drinker(*name*, *address*)
Bar(*name*, *address*)
Frequents(*drinker*, *bar*, *times_a_week*)
Likes(*drinker*, *beer*)
Serves(*bar*, *beer*, *price*)

We would like to represent all this data in one XML file. There are multiple ways to structure the data in a hierarchical manner. We would like to do the following: The document should list all bars first, and then all drinkers. Under a bar element, we also list all beers served at the bar. Under a drinker element, we also list all the beers that the drinker likes, and then all the bars that the drinker frequents. You should capture as many constraints (e.g., keys and foreign keys) as possible. Design an XML Schema for this document, and submit both the XML Schema (`.xsd`) file and an XML (`.xml`) file that represents the data in the sample relational database instance (found in `/opt/dbcourse/examples/db-beers/data/`). You should check your XML file against your XML Schema with command `xmllint` (see "XML Tips" on the course Web site for instructions).