CompSci 316 Fall 2017: Homework #3

100 points (8.75% of course grade) + 10 points extra credit
Assigned: Thursday, October 19
Due: Tuesday, November 7

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, and X1 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 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.

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 begins with "`Susan`" or "`Suzan`". (For each of them, simply print the entire `person` element.) Use `starts-with(str1, str2)` to test if `str1` starts with `str2`.

(b) Find who serves the role of "`Chairman`" on the "`Personnel`" subcommittee of the Senate Committee on Armed Services (code name "`SSAS`"). Simply print the entire `person` element.

(c) List all current Senators who also served in the Congress sometime before 1980. Format each of them as an element of the form `<senator name="…"/>`. Use `xs:date("1939-12-31") < xs:date("2000-01-01")` to test if the date 1939-12-31 precedes the date 2000-01-01.

(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. (By the way, who among them was a professor at Duke?)

(e) 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>`. (By the way, do you know why they aren't?)

(f) Find current legislators who previously served under a different party affiliation. For each such legislator, display his/her current role, followed by previous roles with party affiliation different from the current one. For example (whitespace is unimportant):

```
… <member name="Richard C. Shelby">
    <role current="1" enddate="2023-01-03" party="Republican" startdate="2017-
01-03" state="AL" type="sen"/>
    <role district="7" enddate="1981-01-03" party="Democrat" startdate="1979-
01-15" state="AL" type="rep"/>
    <role district="7" enddate="1983-01-03" …/> …
  </member> …
```

(g) Find the number of current Representatives for each party by gender. Your output should look like the following (whitespace is unimportant):

```
<result>
  <Democrat><M count="…"/><F count="…"/></Democrat>
  <Republican><M count="…"/><F count="…"/></Republican>
  <Independent><M count="…"/><F count="…"/></Independent>
</result>
```

To specify computed values (expressions) as output element/attribute names, you can use the following alternative XQuery syntax for constructing elements/attributes:

```
… return element {$computed_etag} {       (: with {}, tag name is computed :)
    attribute {concat('attr', '1') {$computed_aval},
    attribute attr2 {$computed_aval2},    (: without {}, attr2 becomes the
    …                                        attribute name verbatim :)
  } …
```

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 John McCain:

```
<?xml version="1.0" encoding="UTF-8"?>
<congress>
  <house>
    …
  </house>
  <senate>
    …
    <person name="John McCain">
      <committee name="Senate Committee on Indian Affairs" role="Member"/>
      <committee name="Senate Select Committee on Intelligence" role="Ex Officio"/>
      <committee name="Senate Committee on Armed Services" role="Chairman">
        <subcommittee name="Airland" role="Ex Officio"/>
        <subcommittee name="Emerging Threats and Capabilities" role="Ex Officio"/>
        <subcommittee name="Personnel" role="Ex Officio"/>
        <subcommittee name="Readiness and Management Support" role="Ex Officio"/>
        <subcommittee name="SeaPower" role="Ex Officio"/>
        <subcommittee name="Strategic Forces" role="Ex Officio"/>
        <subcommittee name="Cybersecurity" role="Ex Officio"/>
      </committee>
      <committee name="Senate Committee on Homeland Security and Governmental Affairs" role="Member">
        <subcommittee name="Permanent Subcommittee on Investigations" role="Member"/>
        <subcommittee name="Regulatory Affairs and Federal Management" role="Member"/>
      </committee>
    </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)

In this problem, you will help "wrangle" some semi-structured data into a format that can be managed and analyzed easier. The file `/opt/dbcourse/assignments/hw3/hardware-raw.xlsx` is a Microsoft Excel spreadsheet that Hack Duke 2016 used to track its hardware inventory. Each row of the spreadsheet starts with the name of some hardware, followed by an arbitrary number of codes (one for each actual hardware item). Hardware names are unique, and given one hardware name, the associated codes are distinct. Note that some hardware names contain the string "`(MLH)`", meaning that all such hardware items belong to the Major League Hackathon; otherwise, the hardware items belong to Hack Duke.

(a) Use the programming language/tools of your choice to convert the spreadsheet and convert to a more structured XML format. If you have Microsoft Excel, you can first open `hardware-raw.xlsx` the spreadsheet and save it in CSV or XML format for further processing (for your convenience, we have included the XML version in a file called `hardware-raw.xml` in the same directory). Your code can work from this exported file instead of the original `.xlsx` file.

Name your output file `hardware.xml`. It should be validated against the DTD in `/opt/dbcourse/assignments/hw3/hardware.dtd` and should resemble the following:

```
<?xml version="1.0" encoding="utf-8"?>
<HackDuke2016>
  <hardware name="AmazonBasics 4 Port USB 3.0 Hub" owner="Duke">
    <item code="10352"/>
    <item code="10347"/>
    <item code="10394"/>
    <item code="10352"/>
    <item code="10393"/>
    <item code="10016"/>
    <item code="10017"/>
    <item code="HRD10433"/>
  </hardware>
  <hardware name="Dell XPS Laptop w/ Charger" owner="MLH">
    <item code="00005"/>
    <item code="00004"/>
    <item code="00003"/>
  </hardware>
</HackDuke2016>
```

(b) Design an XML Schema for `hardware.xml` to capture the same formatting constraints as the given DTD, plus additional constraints such as keys. Name your schema file `hardware.xsd`. Make sure your `hardware.xml` conforms to your `hardware.xsd` (using command `xmllint`; see "XML Tips" on the course website for instructions).

Submit your conversion code for (a), its output `hardware.xml`, and the schema file `hardware.xsd`.