Homework 4
XQuery

Due date: Thursday, November 22, 11:59pm

General instructions

This homework tests your knowledge of XQuery, or, to be more exact, facilitates it.

The homework consists of three parts. In part 1, you have to write short XQuery expressions that return desired results — not unlike SQL queries, but applied to XML data. In part 2, you will use XQuery to produce more complex outputs, e.g., format results as veiwable HTML documents. In part 3, you get to play a role of a database designer.

Part 1 expects you to use XQuery as a database query language.

Part 2 expects you to use XQuery as a programming language.

Data

For this homework you will work with two XML files: `menu.xml` and `orders.xml`. The files can be downloaded from the course web page. Links are availble at

http://www.csc.calpoly/edu/∼dekhtyar/560-Fall2006

Direct URLs of the files are:

http://www.csc.calpoly/edu/∼dekhtyar/560-Fall2006/menu.xml
http://www.csc.calpoly/edu/∼dekhtyar/560-Fall2006/orders.xml

menu.xml: This file contains information about the menu of one restaurant. The overall structure of this XML document is:

```
<restaurant>
  <name>A La Lucie</name>
  <location>
     <city>Lexington</city>
     <state>Kentucky</state>
  </location>
  <menu>
    <dish>...</dish>
    ....
    <dish>...</dish>
  </menu>
</restaurant>
```

<restaurant> is the root of the document. It contains three children, <name>, <location> and <menu>.

<name> is the name of the restaurant.

<location> specifies the location of the restaurant in a form of <city>, <state> pair.

<menu> contains information about restaurant's menu. The content of this element is a sequence (list) of <dish> elements.

<dish> elements document information about individual dishes offered by the restaurant.

The general structure of a <dish> element is shown in the following fragment:

```
<dish type="salad">
    <served>lunch</served>
    <served>dinner</served>
    <name>Greek Salad</name>
    <note>Kalamata Olives, Tomatoes, Cucumbers, Onions,
          Bell Peppers, and Mixed Greens. Tossed with Lemon Herb
          Vinaigrette and Feta Cheese. Served with Hummus and Pita.
     </note>
     <price>8.95</price>
</dish>
```

type attribute specifies the category of the dish. The menu contains the following categories: `soup`, `salad`, `appetizer`, `sandwich` and `entree`. Each dish belongs to exactly one category.

<served>. The restaurant is open for lunch and dinner, with separate menus. This element specifies which menu(s) current dish appears on. At least one `<served>` element must be present inside each `<dish>` element. It is possible to have two `<served>` elements. The content of the `<served>` element is either `"lunch"` or `"dinner"`.

<name>: the name of the dish as it appears on the menu. Mandatory.

<note>: notes on the dish as supplied by the chef. Typically, shown in the printed menus to explain what ingredients the dish contains, what it is served with and/or how it is prepared. Optional.

<price>: price of the dish. Mandatory.

<variant>: additional feature of the dish entry (not shown above): described possible variant of the dish. The only variants that appear in the `menu.xml` file have the following structure:

```
<variant>
     <add>Chicken</add>
     <price>8.95</price>
</variant>
```

Here, `<add>` means that the variant of the dish is obtained by adding a specified ingredient (e.g., chicken), which `<price>` shows the full price of the dish with the added ingredient.

orders.xml: this file contains a list of orders for the restaurant described in `menu.xml`. The structure of the XML document is:

```
<orders>
   <restaurant>A La Lucie</restaurant>
   <order> ... </order>
     ...
   <order> ... </order>
</orders>
```

Here,

<orders> is the root of the document. Its first child is `<restaurant>`. Second child and on is the list (sequence) of `<order>` elements.

<restaurant>: name of the restaurant for which the orders are described.

<oder>: encompasses information about one order at the restaurant.

The format of the `<order>` element is seen in the following example:

```
<order id="1">
  <menu>lunch</menu>
  <table>2</table>
  <party>3</party>
  <person seat="1">
     <dish type="soup">Oyster Stew</dish>
     <dish type="salad">Caesar Salad</dish>
  </person>
  <person seat="2">
     <dish type="salad">Chicken Salad Plate</dish>
     <dish type="appetizer">Smoked Salmon</dish>
     <dish type="appetizer">Artichoke and Parmesan Soufflee</dish>
  </person>
  <person seat="3">
     <dish type="salad">Chicken Salad Plate</dish>
     <dish type="sandwich">Sliced Tenderloin</dish>
  </person>
  <payment person="2">joint</payment>
  <tip>0.18</tip>
</order>
```

id attribute of `<order>`. This attribute is the order number - a unique identifier associated with each order.

`<menu>` specifies which of the two menus (lunch or dinner) was used when the customers placed their orders. Mandatory.

`<table>` identifies the specific table in the restaurant's dining room at which the customers who placed current order sat. Optional in general, but is present in every `<order>` object in the file.

`<party>` specifies the number of customers in the current party. Mandatory.

`<person>` contains information about the specific person's food orders.

seat attrbute of `<person>` identifies each person by their position at the dining table.

`<payment>` and its `person` attribute: specifies how the payment of the restaurant check was handled. If its content is `"joint"`, one person at the table (identified by the `person` attribute) has paid the entire bill. If its content is `"separate"`, then each person in the party paid its own check.

`<tip>` element and its `seat` attribute: specifies the amount of tip recieved by the waiter. The amount is specified as a percentage of the check total. If payment was `joint`, then `<tip>` value refers to the entire order. If payment was `separate`, then several `<tip>` elements will appear. Each element will have `seat` attribute, identifying the specific check to which the tip applies.

<dish> element and its type attribute: Each dish element represents a single dish ordered by one person. The type attribute is the same as the type attribute of the <dish> element in menu.xml. The content comes from the <name> child of <dish> from menu.xml.

Debugging

While it is possible to correctly complete this assignemnt without ever touching XQuery-based software (MonetDB, eXist, etc...), it is not very feasible. My suggestion is to use either MonetDB or eXist to debug your queries.

Read in menu.xml and orders.xml. Use the filenames as the internal names for these two documents.

Part 1: Database queries

Write XQuery statements that return the following information:

1. Change the orders.xml file to include price attribute for <dish> elements. Output the result.

2. Find all orders of 3 or more customers at a table. For each order output the entire list of items ordered with notes and prices.

3. Find all lunch orders for table 2.

4. Find all customers who ordered "Smoked Salmon". For each customer, provide their order information, and include in the output the list of *other* dishes they have ordered.

5. Find all customers who ordered entrees that cost more than $20. Output the order and the customer identification, and include the dish name and description.

6. Find all dishes which are served with Mashed Potatoes, output the list of dishes. For each dish, add to its description a new element: <orders> and include inside this element the element <order id="x" seat="y"> for each order and seat that ordered that dish.

Part 2: XQuery programming

1. Write an XQuery program that takes as input orders.xml and menus.xml and for each order generates reciept and computes totals (note that the reciept should contain a 6% sales tax and the tip amount. The tip amount is the specified percentage of the price of an individual order before tax).

2. Write an XQuery program that outputs a nicely formatted HTML version of the lunch menu. All dishes must be categorized by the `type` attrbiute, and listed in the order `soup`, `salad`, `appetizer` `sandwich` and `entree`. The deisgn of the HTML format is left up to you, but the document produced by your query must be readable by a web browser (e.g., Firefox) and must display well.

3. Write an XQuery program that outputs a nicely formatted HTML version of the dinner menu. All dishes must be categorized by the `type` attrbiute, and listed in the order `soup`, `salad`, `appetizer` `sandwich` and `entree`. The deisgn of the HTML format is left up to you, but the document produced by your query must be readable by a web browser (e.g., Firefox) and must display well.

Part 3

I am thinking of making a small XML dataset based on the restaurant theme to be used for training purposes. Your help is solicited.

1. Propose a DTD/Schema for the current version of the dataset (`menu.xml` and `orders.xml`).

2. Propose changes/additions to the current structure of the files in the dataset. What additional information about menu items and/or orders needs to be kept track of? How do you propose to change the DTD/Schema to accommodate for it?

3. Propose any additional files for the dataset. What information is to be stored in the new XML document? What is the proposed DTD/schema for it? What, if any, changed need to be made to the other files in the dataset?