Database Systems - G22.2433 - Spring 2006
Professor: Alberto Lerner – lerner@cs.nyu.edu
Homework 4: Due: ???

Preamble - Install Galax

Galax is a XQuery processor developed by Mary Fernandez and Jerome Simeon, who happen to sit on the W3C XQuery comitee. (This means that they are designers of the language.) It is one of the most complete XQuery implementations around. Yet it is simple to use. Please follow the steps below and you'll be able to install and run this tool in just a couple of minutes.

1. Goto to http://www.galaxquery.org/distrib.html. Pick the binary distribution that suits you best.
2. Open the tar-ball

   ```
   cat Galax-0.5.0-Linux.tar.gz | gunzip | tar xvf -
   ```

3. Put the Galax's bin directory in your PATH

   ```
   export PATH=$PATH:~/Galax-0.5.0-Linux/bin
   ```

4. Now, try the following commands on your prompt ("myprompt>" is not part of the command)

   ```
   myprompt> cat "<two> {1+1} </two>" > test.xq
   myprompt> galax-run test.xq
   ```

   If you get the following output, you're all set! Otherwise send me an email and ask for help.

   ```
   <two>2</two>
   ```

You're going to be using the following files in this exercise:
$(GALAXHOME)/examples/docs/hispo.xml
$(GALAXHOME)/examples/docs/hispo.xsd

Warming-up:
Just output the entire input document. You can define which document you will be working with in the command line

```
myprompt> echo "." > query0a.xq
myprompt> galax-run -context-item hispo.xml query0a.xq
```

You should see the hispo.xml document's contents. Alternatively, you could refer to the document in the expression itself

```
myprompt> echo "doc(\"hispo.xml\")/." > query0b.xq
myprompt> galax-run query0b.xq
```

B - Exercises

Given the document hispo.xml, formulate the following queries using XPath expressions

1. What are all the nodes of this document whose tag is 'item'

```
<item xmlns:HisPo="http://www.hispo.com/" partNum="872-AA">
  <productName>Lawnmower</productName>
  <quantity>1</quantity>
  <USPrice>148.95</USPrice>
  <HisPo:comment>Confirm this is electric</HisPo:comment>
</item>,
<item xmlns:HisPo="http://www.hispo.com/" partNum="926-AA">
  <productName>Baby Monitor</productName>
  <quantity>2</quantity>
  <USPrice>39.98</USPrice>
  <shipDate>1999-05-21</shipDate>
</item>
```

2. What are the items that have already been shipped (that is, that have a shipDate)

```
<item xmlns:HisPo="http://www.hispo.com/" partNum="926-AA">
  <productName>Baby Monitor</productName>
  <quantity>2</quantity>
  <USPrice>39.98</USPrice>
  <shipDate>1999-05-21</shipDate>
</item>
```

3. Show the item whose partNum is 872-AA

```
<item xmlns:HisPo="http://www.hispo.com/" partNum="872-AA">
  <productName>Lawnmower</productName>
  <quantity>1</quantity>
  <USPrice>148.95</USPrice>
  <HisPo:comment>Confirm this is electric</HisPo:comment>
```

```
</item>
```

4. What is the first instance of the node whose tag is 'item'

```
<item xmlns:HisPo="http://www.hispo.com/" partNum="872-AA">
  <productName>Lawnmower</productName>
  <quantity>1</quantity>
  <USPrice>148.95</USPrice>
  <HisPo:comment>Confirm this is electric</HisPo:comment>
</item>
```

5. What is the name of the products that have already been shipped

```
<productName xmlns:HisPo="http://www.hispo.com/">Baby Monitor</productName>
```

6. Same as above, but outputting only the name of the product (without the opening and closing tags

```
text {"Baby Monitor"}
```

7. Find all the items that sold more than one element.

```
<item xmlns:HisPo="http://www.hispo.com/" partNum="926-AA">
  <productName>Baby Monitor</productName>
  <quantity>2</quantity>
  <USPrice>39.98</USPrice>
  <shipDate>1999-05-21</shipDate>
</item>
```

   We will be using more than just XPath in the next query. But try not to use XQuery FLWOR expressions yet. Tip: try a node constuctor.

8. Same as in 7, but now give your answer inside a node tagged 'answer'

```
<answer>
  <item xmlns:HisPo="http://www.hispo.com/" partNum="926-AA">
    <productName>Baby Monitor</productName>
    <quantity>2</quantity>
    <USPrice>39.98</USPrice>
    <shipDate>1999-05-21</shipDate>
  </item>
</answer>
```

Now, use XQuery and FLWOR expressions to formulate the following queries. In XQuery, you can refer to the document in the query, too. For instance,

```
for $d in doc("hispo.xml")
return $d
```

9. Return the name of customer to ship to and the one to bill to

```
<answer>
  <billTo>
    <name xmlns:HisPo="http://www.hispo.com/">Robert Smith</name>
  </billTo>
  <shipTo>
    <name xmlns:HisPo="http://www.hispo.com/">Alice Smith</name>
  </shipTo>
</answer>
```

10. Compute the total sales per partNum

```
<answer>
  <item>
    <partNumber partNum="872-AA"/>
    <totalPurchased>148.95</totalPurchased>
  </item>
  <item>
    <partNumber partNum="926-AA"/>
    <totalPurchased>79.96</totalPurchased>
  </item>
</answer>
```

C - Challenge Question

Add another purchase order to hispo.xml with three items. Let two if these items be the ones in the original purchase order (ie, same partNum and price) but with different quantities. Note that you will need to create a new root node under which the two sub-nodes will reside. Your document should look like the following:

```
<HisPo:purchaseDB>
  <HisPo:purchaseOrder HisPo:orderDate='2006-01-16>
  ... your order with three items
  </HisPo:purchaseOrder>
```

```
  ... the original purchase order
</HisPo:puchaseDB>
```

Create a XQuery query that "inverts the hierachy" purchaseOrder->item. In other words, create a document in which purchaseOrder tags are children of item tags. Your result should look like the following:

```
<invertedDB>
  <items>
    <item partNum="...">
      <purchaseOrder> 1999-10-20 </purchaseOrder>
      <purchaseOrder> 2006-01-16 </purchaseOrder>
    </item>

    <item ...

  </items>
</invertedDB>
```