XML – séance 9

1. Pour chacune des requêtes XQuery suivantes, expliquez ce qu'elle fait et anticipez le résultat de son exécution. Vérifiez à l'aide du logiciel BaseX.

Soit le fichier library.xml suivant :

```
<?xml version="1.0" ?>
<library>
	<songlist>
		<song>
			<track_ID>3485</track_ID>
			<name>Nothing Else Matters</name>
			<artist>Metallica</artist>
			<album>Metallica</album>
			<genre>Metal</genre>
		</song>
		<song>
			<track_ID>3590</track_ID>
			<name>Beat It</name>
			<artist>Michael Jackson</artist>
			<album>Thriller</album>
			<genre>Pop</genre>
		</song>
		<song>
			<track_ID>3597</track_ID>
			<name>Billie Jean</name>
			<artist>Michael Jackson</artist>
			<album>Thriller</album>
			<genre>Pop</genre>
		</song>
	</songlist>
	<playlists>
		<list name="Cool">
			<track_ID>3485</track_ID>
			<track_ID>3590</track_ID>
		</list>
		<list name="MJ">
			<track_ID>3590</track_ID>
			<track_ID>3597</track_ID>
		</list>
	</playlists>
</library>
```

a)
```
<results>
    {for $x in doc("library.xml")//song
    where $x/artist="Michael Jackson"
    return element song{ attribute name{$x/name}}}
  </results>
```

b)
```
<results> {
      for $f in doc("library.xml")//song
      let $s := $f/name,
          $a := $f/artist
      return
         <result>
           {$a}
           {$s}
         </result>
      }
  </results>
```

c)
```
for $x in (doc("library.xml")//list)
   where every $y in ($x/track_ID)
      satisfies (some $s in(doc("library.xml")//song)
            satisfies $s/artist="Michael Jackson" and $s/track_ID=$y)
   return <liste>{$x/@name}</liste>
```

d)
```
for $x in distinct-values(doc("library.xml")//artist)
   return element artiste {
      attribute name{$x},element songlist{
        for $y in (doc("library.xml")//song[artist=$x])
        return element song {attribute name{$y/name}}
      }
   }
```

e)
```
(   for $x in distinct-values(doc("library.xml")//genre)
      let $c:= count (doc("library.xml")//song[genre=$x])
      order by $c ascending
      return
      <genre>
            <name>{$x}</name>
            <nombre>{$c}</nombre>
      </genre>
   )[last()]
```

2. Voici un second fichier duree.xml (ce document XML donne les durées des chansons en seconde)

```
<?xml version="1.0" ?>
<durees>
	<song track_ID="3485">
		<duree>345</duree>
	</song>
	<song track_ID="3590">
		<duree>312</duree>
	</song>
	<song track_ID="3597">
		<duree>242</duree>
	</song>
</durees>
```

a) Ecrivez une requête qui renvoie l'artiste et le titre de la chanson la plus longue. La sortie devrait ressembler à cela :

```
<?xml version="1.0" encoding="UTF-8"?>
<song><artist>Metallica</artist><name>Nothing Else Matters</name></song>
```

b) Ecrivez une requête qui affiche toutes les informations d'une chanson (à savoir les infos de library.xml ainsi que la durée contenue dans duree.xml). La sortie devrait ressembler à cela :

```
<?xml version="1.0" encoding="UTF-8"?>
<songlist>
	<song>
		<track_ID>3485</track_ID>
		<name>Nothing Else Matters</name>
		<artist>Metallica</artist>
		<album>Metallica</album>
		<genre>Metal</genre>
		<duree>345</duree>
	</song>

	…

</songlist>
```

Indice : node() permet d'obtenir tous les enfants d'un noeud.

c) Ecrivez une requête qui calcule la durée totale de chaque liste contenue dans library.xml. La sortie devrait ressembler à cela :

```
<?xml version="1.0" encoding="UTF-8"?>
<playlist>
	<list name="Cool">
		<duree>657</duree>
	</list>
	<list name="MJ">
		<duree>554</duree>
	</list>
</playlist>
```

d) Même question que la précédente mais en affichant uniquement les listes qui durent plus de 600 secondes.

Il est possible de répondre à cette question de plusieurs manières :

- tentez votre chance en adaptant uniquement le xpath de l'expression précédente
- essayez aussi en utilisant une expression conditionnelle

  Indice : http://stackoverflow.com/questions/3687218/how-to-use-if-else-in-xquery-assignment