XML – séance 9/correctif

Soit le fichier 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>
```

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.

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

```
(: Tous les noms de chansons de Michael Jackson:)
<?xml version="1.0" encoding="UTF-8"?>
```

```
<results>
      <song name="Beat It" />
      <song name="Billie Jean" />
</results>
```

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

```
(: tous les couples artistes-chansons :)
```

```
<?xml version="1.0" encoding="UTF-8"?>
<results>
      <result>
            <artist>Metallica</artist>
            <name>Nothing Else Matters</name>
      </result>
      <result>
            <artist>Michael Jackson</artist>
            <name>Beat It</name>
      </result>
      <result>
            <artist>Michael Jackson</artist>
            <name>Billie Jean</name>
      </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>

(: toutes les noms de list qui ne contiennent que des chansons de MJ:)
<?xml version="1.0" encoding="UTF-8"?><liste name="MJ"/>

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}}
      }
   }

(: Renvoie chaque artiste ainsi que ses titres :)

<?xml version="1.0" encoding="UTF-8"?>
```

```
<artiste name="Metallica">
	<songlist>
		<song name="Nothing Else Matters" />
	</songlist>
</artiste>
<artiste name="Michael Jackson">
	<songlist>
		<song name="Beat It" />
		<song name="Billie Jean" />
	</songlist>
</artiste>
```

```
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()]

(: renvoie le genre où il y a le plus de chanson  :)
```

```
<?xml version="1.0" encoding="UTF-
8"?><genre><name>Pop</name><nombre>2</nombre></genre>
```

2) Soit le fichier duree.xml suivant (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>
```

```
let $x:=doc("duree.xml")//song[duree=max(doc("duree.xml")//song/duree)],
    $y:=doc("library.xml")//song[track_ID=$x/@track_ID]
return <song>{$y/artist,$y/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>
```

```
<songlist>
    {for $x in (doc("library.xml")//song)
    let $c:= doc("duree.xml")//song[@track_ID=$x/track_ID]/duree
    return <song>{$x/node(),$c}</song>
    }
</songlist>
```

c) Ecrivez une requête qui calcule la durée totale de chaque list 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>
```

```
<playlist>
    {for $l in (doc("library.xml")//list)
    let $c:= sum(doc("duree.xml")//song[@track_ID=$l/track_ID]/duree
    )
    return element list{$l/@name,element duree{$c}}
    }
</playlist>
```

(:version em:)

```
<playlist>
{for $x in (doc("library.xml")//list)
let $tot:=sum(doc("duree.xml")//song[@track_ID=$x/track_ID]/duree)
return <list>{$x/@*,<duree>{$tot}</duree>}</list>
}
</playlist>
```

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

```
<playlist>
{for $x in (doc("library.xml")//list)
let $tot:=sum(doc("duree.xml")//song[@track_ID=$x/track_ID]/duree)
return <list>{$x/@*,<duree>{$tot}</duree>}</list> [duree>=600]
}
```

Ou

```
<playlist>
{for $x in (doc("library.xml")//list)
let $tot:=sum(doc("duree.xml")//song[@track_ID=$x/track_ID]/duree)
return if($tot>=600)then<list>{$x/@*,<duree>{$tot}</duree>}</list> else ()
}
</playlist>
```