Corrigé de l'exercice 2 du TD d'interrogation par XQuery de la base "mondial.xml3 (TD 4), commenté.
En général, plusieurs solutions sont possibles, donc le corrigé n'est pas exhaustif.

1) Pays de plus que 6000000 habitants.

```
for $p in doc("mondial.xml")/mondial/country let $b:=$p/population where
$b >6000000 return $p/name.
```

Aussi :

```
for $p in doc("mondial.xml")//country  where
$p/population >6000000 return $p/name.
```

Et encore :

```
doc("mondial.xml")//country[population>6000000]/name
```

NB : Cette dernière solution n'utilise pas de variables :
c'est une expression pure de XPATH (XQUERY est une extension de XPATH).

Bien comprendre pourquoi les 3 solutions proposées calculent exactement la même chose.

2) Codes des pays bordant la France ?

Une première approximation :

```
doc("mondial.xml")//country[name="France"]/border
```

La formulation ci-dessus donne TROP d'information : on ne veut que les CODES des pays. Une solution consiste à utiliser le mot-clé data :

```
for $p in doc("mondial.xml")//country[name="France"]/border return <code_pays>
{data($p/@country)} </code_pays>
```

Le mot-clé data permet d'extraire la valeur (« données » = data) de l'attribut country de (la valeur de) $p.

On observera aussi que la seconde occurrence du mot ``country'' est utilisée comme ATTRIBUT de l'élément border, tandis que la première indique un fils de l'élément mondial : étudier le DTD de

la base mondial. Dans ce DTD, le mot ``country'' est sur-chargé.

Remarquer aussi que l'attribut country de border est de type IDREF : il sert
donc à pointer sur l'identificateur (la valeur de l'attribut car_code, qui est de type ID) d'un pays.

Enfin : ça peut sembler étrange que l'attribut country de border soit placé
après un /, comme si c'était un fils de la valeur de la
variable $p, mais ça marche ainsi, car, Xquery « voit » les attributs comme des noeuds particuliers,
(voir le cours sur Xquery à ce propos).

3)
Nombre de provinces en France ?

On utilise ici le mot-clé count qui sert à compter (comme en SQL !) :

count(doc("mondial.xml")//country[name="France"]/province)

4) Caractéristiques de Paris ?

On veut ici toutes les informations possibles sur Paris.

doc("mondial.xml")//city[name="Paris"]

Attention : les city sont des fils de province, pas de country.

5) Le nom et la population de la 3ème ville de France dans
l'ordre du document.

Déjà :

doc("mondial.xml")//country[@car_code="F"]//city[3]

donne l'entière description de la 3ème ville de France
DANS L'ORDRE DU DOCUMENT (construction [n], où n est un entier positif :
vue en cours).

Mais cela donne trop (toute la description de la ville). On peut faire :

doc("mondial.xml")//country[@car_code="F"]//city[3]/(name, population)

On remarquera ici l'utilisation du constructeur « ( ) » de liste de
balises : (bal1,...,balN).
Observer la différence de la réponse si l'on écrit :

doc("mondial.xml")//country[@car_code="F"]//city[3]/(population, name)

6) En utilisant les constructeurs, récrire le fichier pour obtenir l'élement
suivant pour chaque pays :

<pays code = ??>
    <nom> ?? </nom>
    <population> ?? <population>

Solution (une des solutions possibles) :

for $x in doc("mondial.xml")//country
return element pays {
attribute code {$x/@car_code},
element nom {$x/name/text()},
$x/population}

Observer :

1) l'utilisation des constructeurs :
attribute NOM-ATTRIBUT CONTENU
element NOM-ELEMENT CONTENU
pour créer, respectivement, un nouveau élément et un nouveau attribut,
2) l'utilisation de la fonction text() pour extraire le contenu (textuel) de l'ancien élément
« name » du fichier sans reproduire la balise "name". Pour comprendre
ceci, essayer la variante suivante, sans text :

for $x in doc("mondial.xml")//country
return element pays {
attribute code {$x/@car_code},
element nom {$x/name},
$x/population}

Variante avec le mot-clé data à la place de text :

for $x in doc("mondial.xml")//country
return element pays {

attribute code {$x/@car_code},
element nom {data($x/name)},
$x/population}

7) Générer les éléments suivants pour la France :

<ville position = ??> ?? </ville>

La position, ici, est la position dans le document.

Ici, on utilise le nouveau mot clé at (voir le cours) :
at $i fait varier la variable $i sur toutes les
positions (de l'élément considérés) dans le document.

for $a at $i in doc("mondial.xml")//country//city
return element ville {attribute position {$i}, $a/name/text()}

Oberver aussi, à nouveau, l'utilisation de text() pour extraire le continu des éléments nommés name dans le fichier d'origine et celle des constructeurs element et attribute
(déjà expliqués dans le commentaire à la question 6).

Une variante de la même solution, avec data à la place de text :

for $a at $i in doc("mondial.xml")//country//city return element ville {attribute position {$i},
data($a/name)}

8) Pour chaque pays, générer l'élément suivant :
<pays nom = ??> <villes> ville 1 ville 2 ... </villes>
(avec les trois points)
si le nombre de villes est supérieur à 2, sinon
<pays nom = ??> <villes> ville 1 ville 2 </villes>

Ici, on utilise le if ... then ... else de XQuery

for $a in doc("mondial.xml")//country return element pays {
attribute nom {$a/name/text()},
element villes {
for $v at $i in $a//city where $i <= 2 return
(string($v/name)),
if (count($a//city) > 2) then "..." else ""}}

Observer l'utilisation du constructeur string.
Pour comprendre pourquoi il est nécessaire, ici, tester la
variante sans string :

```
for $a in doc("mondial.xml")//country return element pays {
attribute nom {$a/name/text()},
element villes {
for $v at $i in $a//city where $i <= 2 return
$v/name,
if (count($a//city) > 2) then "..." else ""}}
```

L'affichage qui en résulte est différent, et ne respecte
pas la spécification.

9) Toutes les religions, sans doublons.

Mot clé utile : distinct-values.

```
distinct-values(doc("mondial.xml")//religions)
```

Tester la différence avec :
doc("mondial.xml")//religions

10) Noms des pays bordant la France

La différence entre cette requête, et la 2, c'est que la 2 demandait les **codes** des pays bordant la France, tandis qu'ici on demande les **noms**. Or, l'attribut country de border (qui est de type IDREF) ne donne que les codes (qui sont les identificateurs des pays). Donc il faut faire une sort de jointure :
a) on recupère les codes des pays bordant la France (comme pour la 2)
b) puis on sélectionne les noms des pays ayant ces codes.

Voici une solution, qui utilise deux for :

```
for $a in doc("mondial.xml")//country[name ="France"]/border/@country
for $b in doc("mondial.xml")//country
where $b/@car_code = $a return $b/name/text()
```

11) Donner les pays frontaliers communs à l'Allemagne et
à la France.

Avec cette solution on obtient les codes des pays :

```
for $a in doc("mondial.xml")//country[@car_code="D"]/border/@country
for $f in doc("mondial.xml")//country[@car_code="F"]/border/@country
where $a=$f return data($f)
```

12) C'est comme la 11.

13) Combien de fois la superficie de la France fait celle du Luxenbourg ?

```
let $a:=doc("mondial.xml")//country[name="France"]/@area
 let $b:=doc("mondial.xml")//country[name="Luxembourg"]/@area
 return $a div $b
```

14) Combien d'habitants au km carré en France ?

```
let $a:=doc("mondial.xml")//country[name="France"] return $a/population div
$a/@area
```

15)Quelle est la superficie de l'Europe ?
Remarque : la solution suivante est pour la version de mondial.xml3qui contient seulement les pays européens :

```
let $a:=doc("mondial.xml")//country/@area
return sum($a)
```

Remarquer l'action de la fonction pre-définie sum est de sommer
les PLUSIEURS valeurs que $a reçoit; pour s'en rendre compte,
essayer:

```
data(doc("mondial.xml")//country/@area)
```

qui nous donne, justement, toutes les valeurs possibles
pour l'attribut area.

16) Les plus petit pays est-il aussi le moins peuplé ?
Attention, c'est probablement la requête la plus dure de la feuille.

```
let $a1:= for $a in doc("mondial.xml")//country order by ($a/@area) descending
return $a/name
let $a2:= for $a in doc("mondial.xml")//country order by ($a/population) descending
```

return $a/name
return $a1[last()] is $a2[last()]

Si is ne marche pas, essayer avec =

A remarquer : ici la variable $a1 reçoit une valeur qui est
le résultat d'une sous-requête for, c'est-à-dire :
for $a in doc("mondial.xml")//country order by ($a/@area) descending
return $a/name et on a une chose semblable avec $a2.
La syntaxe de Xquery permet cela.

L'idée est donc : $a1 reçois la suite des noms des pays, ordonnée en ordre
décroissant par rapport à la surface. Donc le dernier de la liste est le
pays le plus petit.
La variable $a1 reçois la suite des noms des pays, ordonnée en ordre
décroissant par rapport à la population. Donc le dernier de la liste est le
pays le moins peuplé.
Donc, puisque on a :
return $a1[last()] is $a2[last()]
on va avoir la réponse True si et seulement si le pays le plus petit est
aussi le moins peuplé.

17) La Slovénie est plus peuplée que la Slovaquie ?

let $a:=doc("mondial.xml")//country order by $a descending
return $a[name="Slovakia"]<<$a[name="Slovenia"]

18) Union des pays frontaliers de la France et de l'Espagne

let $u := distinct-values(doc("mondial.xml")//country[name="Germany"]/border/@country union
doc("mondial.xml")//country[name="Spain"]/border/@country)
return $u