MIF 04 GDW - TD 1

Exercice 1:

Pour chaque élément du document XML ci-dessous, donner son espace de nommage.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<categorie xmlns="http://macollection.com">
  <nom >
    Categorie <em xmlns="http://www.w3.org/1999/xhtml">A</em></nom>
  <categorie>
    <nom >B</nom>
    <categorie xmlns:xhtml="http://www.w3.org/1999/xhtml">
      <nom ><xhtml:em >C</xhtml:em></nom>
    </categorie>
  </categorie>
  <coll:categorie xmlns="http://www.w3.org/1999/xhtml"
         xmlns:coll="http://macollection.com">
    <coll:nom ><b >D</b></coll:nom>
    <categorie >
      <coll:nom >E</coll:nom>
    </categorie>
  </coll:categorie>
</categorie>
```

Exercice 2: Schemas

On considère la DTD suivante :

```
<!DOCTYPE inscriptions [
<!ELEMENT inscriptions (ue+,etudiant*)>
<!ELEMENT ue (titre, resume, inscrit*)>
<!ATTLIST ue code ID  #REQUIRED
             niveau CDATA #REQUIRED>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT resume (#PCDATA)>
<!ELEMENT inscrit EMPTY>
<!ATTLIST inscrit num IDREF #REQUIRED
                  semestre CDATA #REQUIRED>
<!ELEMENT etudiant (nom,adresse)>
<!ATTLIST etudiant num ID #REQUIRED>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT adresse (#PCDATA)>
]>
```

On considère également le schéma XML suivant :

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

```

```
3       <xsd:element name="inscriptions">
4           <xsd:complexType>
5               <xsd:sequence>
6                   <xsd:element name="ue" maxOccurs="unbounded"
7                                type="ueContent"/>
8                   <xsd:element name="etudiant" maxOccurs="unbounded"
9                                minOccurs="0" type="etudiantContent"/>
10              </xsd:sequence>
11          </xsd:complexType>
12      </xsd:element>
13
14      <xsd:complexType name="ueContent">
15          <xsd:sequence>
16              <xsd:element name="titre" type="xsd:string"/>
17              <xsd:element name="resume" type="xsd:string"/>
18          </xsd:sequence>
19          <xsd:attribute name="code" type="xsd:ID" use="required"/>
20          <xsd:attribute name="niveau" type="xsd:string" use="required"/>
21      </xsd:complexType>
22
23      <xsd:complexType name="etudiantContent">
24          <xsd:sequence>
25              <xsd:element name="nom" type="xsd:string"/>
26              <xsd:element name="adresse" type="xsd:string"/>
27              <xsd:element name="inscription" minOccurs="0"
28                           maxOccurs="unbounded">
29                  <xsd:complexType>
30                      <xsd:attribute name="codeUe" type="xsd:IDREF"
31                                     use="required"/>
32                      <xsd:attribute name="semestre" type="xsd:string"
33                                     use="required"/>
34                  </xsd:complexType>
35              </xsd:element>
36          </xsd:sequence>
37          <xsd:attribute name="num" type="xsd:ID" use="required"/>
38      </xsd:complexType>
39
40  </xsd:schema>
```

Soit D_{DTD} l'ensemble des documents valides par rapport à la DTD et D_{XSD} l'ensemble des documents valides par rapport au schéma XML.

1. Donner un exemple de document valide par rapport à D_{DTD}, puis un autre pour D_{XSD}.
2. Donner une représentation de la DTD et du XML Schema sous forme de grammaires d'arbre régulières.
3. Caractériser D_{DTD} et D_{XSD}: Ces deux ensembles sont ils disjoints? L'un des deux ensembles contient-il l'autre? Ces deux ensembles sont-ils égaux?
4. Donner un schéma relationnel permettant de stocker l'information contenue dans un fichier de D_{DTD}.

Pour les questions suivantes, on pourra utiliser l'utilitaire en ligne de commande XQilla[1]. Pour les requêtes XPath on pourra aussi utiliser `xmllint`[2]. Des exemples de documents conformes à

[1] Installation Linux: package, MacOSX: via Homebrew, Windows: via Chocolatey ou dans un Linux WSL
[2] Linux: package libxml, MacOSX: préinstallé avec XCode, Windows: uniquement WSL

D_{DTD} sont fournis sur la page du cours.

5. En supposant que l'on interroge un document de D_{DTD}, répondre aux questions suivantes à l'aide de requêtes XPath:
   (a) Donner les numéros d'étudiants.
   (b) Quelle est la chaine de texte (inner text) du nom de l'étudiant dont le numéro est 123456 ?
   (c) Donner les codes des UEs.
   (d) Donner l'UE dont le code est MIF03.
   (e) Donner les ID des inscrits à MIF03.
   (f) Quels sont les noms des étudiants inscrits à MIF03 ?
6. Donner une requête XQuery permettant de passer d'un document de D_{DTD} à un document de D_{XSD}. On pourra commencer par écrire une première version de la requête qui ne prend pas en compte les inscriptions avant d'écrire la requête complète.
7. Y a-t-il perte d'information?

Exercice 3: Fonctions XQuery

Dans cet exercice, on considèrera des documents conformes au schéma suivant:

```
<!DOCTYPE collection
[
<!ELEMENT collection (artist*)>
<!ELEMENT artist (name,album+)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT album (title?,track+)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT track (title)>
<!ATTLIST track seconds CDATA #REQUIRED>
]>
```

Il est possible de définir des fonctions dans XQuery de la manière suivante:

```
declare function nom($arg1 as type1, $arg2 as type2, ...) as type_retour
{ corps de la fonction };
```

Le nom de la fonction doit être préfixé. Il existe un préfixe créé par défaut: local.

Les types peuvent être:

- `text()`, `comment()`, `processing-instruction() element()`, `attribute()` - le nom peut être spécifié entre les parenthèses
- `xs:type`, où type est un type défini dans la norme XML Schema, par exemple `xs:string`, `xs:boolean`, `xs:number`, `xs:integer`.

Un type peut être suivit de *, + ou ? afin d'exprimer le fait de pouvoir gérer une séquence d'éléments au lieu d'un seul élément. Les déclaration de type sont facultatives.

Il faut également noter qu'une fonction ne possède pas point de départ dans l'arbre XML. Il faut donc lui passer un argument qui servira de point de départ lorsque cela est nécessaire.

Exemple de fonction:

```
declare function
local:possede_titre($art as element(artist),$song as xs:string)
{
  $art/album[track/title=$song]
};
```

La fonction s'utilise ensuite de manière classique:

```
local:possede_titre(//artist,'b')
```

Écrire une fonction XQuery qui, étant donné un artiste, renvoie l'ensemble des pistes (track) classées par durée (à durée égale, on comparera selon le titre du morceau, puis selon le titre de l'album).

On suppose à présent que l'on travaille sur un document correspondant à la DTD suivante:

```
<!ELEMENT categorie (nom,description?,sous-categorie*,categorie*)>
<!ATTLIST categorie id CDATA #IMPLIED>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT sous-categorie EMPTY>
<!ATTLIST sous-categorie id CDATA #REQUIRED>
```

On suppose également que l'attribut `id` d'un élément `sous-categorie` correspond à l'attribut `id` d'un élément `categorie` du même document.

Écrire une fonction XQuery qui, étant donné un noeud correspondant à un élément `categorie` réécrit cet élément en remplaçant récursivement les éléments `sous-categorie` par les éléments `categorie` qui leur correspondent. Le but est ainsi d'obtenir un élément `categorie` ayant la même signification, mais sans élément `sous-categorie`.