MIF 04 GDW - TD 1
Correction

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>
```

Correction: `++` ↔ `"http://macollection.com"`, `**` ↔ `"http://www.w3.org/1999/xhtml"`

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">

    <xsd:element name="inscriptions">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="ue" maxOccurs="unbounded"
                             type="ueContent"/>
                <xsd:element name="etudiant" maxOccurs="unbounded"
                             minOccurs="0" type="etudiantContent"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:complexType name="ueContent">
        <xsd:sequence>
            <xsd:element name="titre" type="xsd:string"/>
            <xsd:element name="resume" type="xsd:string"/>
        </xsd:sequence>
        <xsd:attribute name="code" type="xsd:ID" use="required"/>
        <xsd:attribute name="niveau" type="xsd:string" use="required"/>
    </xsd:complexType>

    <xsd:complexType name="etudiantContent">
        <xsd:sequence>
            <xsd:element name="nom" type="xsd:string"/>
            <xsd:element name="adresse" type="xsd:string"/>
            <xsd:element name="inscription" minOccurs="0"
                         maxOccurs="unbounded">
                <xsd:complexType>
                    <xsd:attribute name="codeUe" type="xsd:IDREF"
                                   use="required"/>
                    <xsd:attribute name="semestre" type="xsd:string"
                                   use="required"/>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
        <xsd:attribute name="num" type="xsd:ID" use="required"/>
    </xsd:complexType>

</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}.

   Correction: D_{DTD}:

```
<?xml version="1.0"?>
<!DOCTYPE inscriptions SYSTEM "inscriptions.dtd">
<inscriptions>
  <ue code="BDAV" niveau="4">
    <titre>Bases de donnees avancees</titre>
```

```
    <resume>
      XML, Datalog, Securite
    </resume>
    <inscrit num="123456" semestre="2010/Automne"/>
  </ue>
  <etudiant num="123456">
    <nom>Toto</nom>
    <adresse>Villeurbanne</adresse>
  </etudiant>
</inscriptions>
```

D_{XSD}:

```
<?xml version="1.0"?>
<inscriptions>
  <ue code="BDAV" niveau="4">
    <titre>Bases de donnees avancees</titre>
    <resume>
      XML, Datalog, Securite
    </resume>
  </ue>
  <etudiant num="123456">
    <nom>Toto</nom>
    <adresse>Villeurbanne</adresse>
    <inscription codeUe="BDAV" semestre="2010/Automne"/>
  </etudiant>
</inscriptions>
```

2. Donner une représentation de la DTD et du XML Schema sous forme de grammaires d'arbre régulières.

   Correction:
   DTD:

   $I \rightarrow$ `inscriptions` () ($U+$,E^*)
   $U \rightarrow$ `ue` ((`code`,id,1),(`niveau`,string,1)) (T,R,ER^*)
   $ER \rightarrow$ `inscrit` ((`num`,idref,1),(`semestre`,string,1)) ()
   $E \rightarrow$ `etudiant` ((`num`,id,1)) (N,A)
   $T \rightarrow$ `titre` () (St)
   $R \rightarrow$ `resume` () (St)
   $N \rightarrow$ `nom` () (St)
   $A \rightarrow$ `adresse` () (St)
   $St \rightarrow$ string

   XSD:

   $I \rightarrow$ `inscriptions` () ($U+$,E^*)
   $U \rightarrow$ `ue` ((`code`,id,1),(`niveau`,string,1)) (T,R)
   $E \rightarrow$ `etudiant` ((`num`,id,1)) (N,A,UR^*)
   $UR \rightarrow$ `inscription` ((`codeUe`,idref,1),(`semestre`,string,1)) ()
   $T \rightarrow$ `titre` () (St)
   $R \rightarrow$ `resume` () (St)
   $N \rightarrow$ `nom` () (St)
   $A \rightarrow$ `adresse` () (St)
   $St \rightarrow$ string

3. Caractériser D_{DTD} et D_{XSD}: Ces deux ensembles sont ils disjoints?

Correction: non: un document sans inscription est valide par rapport aux deux schemas
L'un des deux ensembles contient-il l'autre?

Correction: non: s'i y a une inscription, elle n'est pas valable dans l'autre schéma
Ces deux ensembles sont-ils égaux?

Correction: non, sinon ils se contiendraient mutuelllement

4. Donner un schéma relationnel permettant de stocker l'information contenue dans un fichier de D_{DTD}.

   Correction:

```
CREATE TABLE UE(code VARCHAR(255) PRIMARY KEY,
                niveau VARCHAR(255),
                titre VARCHAR(255),
                resume TEXT);
CREATE TABLE ETUDIANT(num INTEGER PRIMARY KEY,
                      nom VARCHAR(255),
                      adresse TEXT);
CREATE TABLE INSCRIPTION(num INTEGER REFERENCES ETUDIANT(num),
                         code VARCHAR(255) REFERENCES UE(code),
                         semestre VARCHAR(255),
                         PRIMARY KEY (num,code,semestre));
```

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 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.
   Correction: `//etudiant/@num`
   Commande: `xmllint --xpath "//etudiant/@num" InscriptionsDTD.xml`

   (b) Quelle est la chaine de texte (inner text) du nom de l'étudiant dont le numéro est 123456 ?
   Correction: `//etudiant[@num=':123456']/nom/text()`
   Commande:
```
xmllint --xpath "//etudiant[@num=':123456']/nom/text()" \
   InscriptionsDTD.xml
```

   (c) Donner les codes des UEs.
   Correction: `//ue/@code`
   Commande: `xmllint --xpath "//ue/@code" InscriptionsDTD.xml`

   (d) Donner l'UE dont le code est MIF03.
   Correction: `//ue[@code='MIF03']`
   Commande: `xmllint --xpath "//ue[@code='MIF03']" InscriptionsDTD.xml`

   (e) Donner les ID des inscrits à MIF03.
   Correction: `//ue[@code='MIF03']/inscrit/@num`
   Commande:
```
xmllint --xpath "//ue[@code='MIF03']/inscrit/@num" \
   InscriptionsDTD.xml
```

[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

(f) Quels sont les noms des étudiants inscrits à MIF03 ?

Correction: `//etudiant[@num=//ue[@code='MIF03']/inscrit/@num]/nom/text()`
Commande:

```
xmllint --xpath \
    "//etudiant[@num=//ue[@code='MIF03']/inscrit/@num]/nom/text()" \
    InscriptionsDTD.xml
```

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.

   Correction:

```
<inscriptions>
  {
    for $ue in /inscriptions/ue
    return
    <ue>{$ue/@*}
    {$ue/titre}
    {$ue/resume}
    </ue>
  }
  {
    for $etu in /inscriptions/etudiant
    return
    <etudiant>{$etu/@*}
    {$etu/*}
    {
      for $i in //inscrit[@num=$etu/@num]
      return
        <inscription codeUe="{$i/../@code}">{$i/@semestre}</inscription>
    }
    </etudiant>
  }
</inscriptions>
```

7. Y a-t-il perte d'information?

   Correction: Non, car il est possible d'écrire un programme XQuery faisant la transformation inverse et obtenir alors un document identique (à l'ordre entre les étudiants, UEs et inscrits près).

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).
Correction:

```
declare function local:track_by_duration($art as element(artist))
as element(track)*
{
  for $tr in $art/album/track
  order by $tr/@seconds, $tr/title, $tr/../title
  return $tr
};
```

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`.

Correction:

```
declare function local:expand-cat($cat as element(categorie))
as element(categorie)
{
 <categorie>
    {$cat/nom}
    {$cat/description}
    {
      for $sc in $cat/sous-categorie
      let $id := $sc/@id
      return
        local:expand-cat(
          $sc/ancestor-or-self::document-node()//categorie[@id = $id])
    }
    {
      for $c in $cat/categorie
      return local:expand-cat($c)
    }
  </categorie>
};

local:expand-cat(/categorie)
```