

Temps total autorisé : 1h30

Les matériaux imprimés (notes de cours ou de TD/TP) sont autorisés. Les téléphones et les ordinateurs sont interdits. La présentation sera prise en compte pour la correction des exercices. Les poids approximatifs sont donnés en parenthèses.

Exercice 1. DTD (4 points)

Donnez un schéma DTD qui valide des documents de la même nature que le document `entraîneurs.xml` en Figure 1. Plus concrètement, votre DTD doit valider des documents qui ont :

1. Un élément racine `entraîneurs`, qui contient un nombre arbitraire de fils `entraîneur` (mais au moins un).
2. Un élément `entraîneur` qui a un fils `nom`, suivi par une séquence d'éléments `club` et `nationale` (en nombre arbitraire, mais au moins un. Il peut également y avoir un nombre arbitraire de `nationale`). Il a aussi un attribut obligatoire `id` de type ID.
3. Un élément `club` a un fils obligatoire `debut`, suivi par un fils optionnel `fin`. Il a également un attribut obligatoire `nom`.
4. Un élément `nationale` doit avoir exactement le même modèle de contenu que l'élément `club` sauf pour le nom de `nationale` qui ne sera pas de type ID.
5. Tous les autres éléments (`nom`, `debut`, `fin`) doivent avoir un contenu textuel.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE entraîneurs SYSTEM "entraîneurs.dtd">
<entraîneurs>
  <entraîneur id="e1">
    <nom>Rudi Garcia</nom>
    <club nom="LOSC"><debut>2008</debut><fin>2013</fin></club>
    <club nom="OM"><debut>2016</debut></club>
  </entraîneur>
  <entraîneur id="e2">
    <nom>Marcelo Bielsa</nom>
    <nationale nom="Argentine"><debut>1998</debut><fin>2004</fin></nationale>
    <club nom="OM"><debut>2014</debut><fin>2015</fin></club>
    <club nom="LOSC"><debut>2017</debut></club>
  </entraîneur>
  <entraîneur id="e3">
    <nom>Didier Deschamps</nom>
    <club nom="OM"><debut>2009</debut><fin>2012</fin></club>
    <nationale nom="France"><debut>2012</debut></nationale>
  </entraîneur>
</entraîneurs>
```

FIGURE 1 – entraîneurs.xml

Exercice 2. XML Schema (4 points) Le squelette d'un fichier XML Schema est donné en Figure 2. Donnez le code XML Schema qui manque afin de créer un schéma qui valide le document de la Figure 1. Plus précisément, il faut ajouter :

1. Le type simple `typeAnnee`, qui définit des chaînes de 4 chiffres.

2. Le type complexe `typeMandat`, qui est utile pour définir un mandat d'un entraîneur à un club ou à une nationale. Le type `typeMandat` a un fils obligatoire `debut`, suivi par un fils optionnel `fin` et un attribut obligatoire `nom`.
3. Le contenu de l'élément racine `entraîneurs`, qui contient un nombre arbitraire de fils `entraîneur` (mais au moins un).

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:simpleType name="typeAnnee">
    <!-- A COMPLETER -->
  </xs:simpleType>

  <xs:complexType name="typeMandat">
    <!-- A COMPLETER -->
  </xs:complexType>

  <xs:complexType name="typeEntraîneur">
    <xs:sequence>
      <xs:element name="nom" type="xs:string"/>
      <xs:choice maxOccurs="unbounded">
        <xs:element name="club" type="typeMandat"/>
        <xs:element name="nationale" type="typeMandat"/>
      </xs:choice>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string"/>
  </xs:complexType>

  <xs:element name="entraîneurs">
    <xs:complexType>
      <!-- A COMPLETER -->
    </xs:complexType>
  </xs:element>
</xs:schema>
```

FIGURE 2 – squelette.xsd

Exercice 3. XPath (4 points)

Question 1. Prenons les deux requêtes XPath suivante :

Q_1 : `count(//entraîneur[./club/@nom="OM"])`

Q_2 : `count(//entraîneur[./club/@nom="OM"])`

- (a) Expliquez le résultat de chacune des deux requêtes sur le document de la Figure 1.
- (b) Est-ce que les deux requêtes donnent toujours le même résultat ? Si oui, expliquez pourquoi. Sinon, donnez un exemple de document sur lequel elles donnent des réponses différentes.

Question 2. Donnez les requêtes XPath pour retourner :

- (a) Les noms des entraîneurs qui sont actuellement en train d'entraîner une équipe nationale.
Indice : ils ont un fils `nationale` qui n'a pas de fils `fin`.
- (b) Le nom de l'entraîneur actuel de LOSC.

Exercice 4. XSLT (4 points)

Question 1. Dessinez le tableau HTML qui sera affiché suite à l'exécution de la transformation donnée en Figure 3, en ayant en entrée le document de la Figure 1.

Question 2. Modifiez la transformation pour que le tableau HTML résultat ne contienne pas les lignes où on a des cellules vides pour `fin`. Il est inutile d'écrire toute la nouvelle transformation : il suffit donc d'indiquer quelle ligne de la transformation il faut modifier et comment vous voulez la modifier.

```

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html"/>

  <xsl:template match="/">
    <html>
      <body>
        <table border="1">
          <tr>
            <th>entraîneur</th>
            <th>club</th>
            <th>debut</th>
            <th>fin</th>
          </tr>
          <xsl:apply-templates select="//entraîneur"/>
        </table>
      </body>
    </html>
  </xsl:template>
  <xsl:template match="entraîneur">
    <xsl:apply-templates select="./club"/>
  </xsl:template>

  <xsl:template match="club">
    <tr>
      <td><xsl:value-of select="../nom"/></td>
      <td><xsl:value-of select="./@nom"/></td>
      <td><xsl:value-of select="debut"/></td>
      <td><xsl:value-of select="fin"/></td>
    </tr>
  </xsl:template>
</xsl:stylesheet>

```

FIGURE 3 – stylesheet.xml

Exercice 5. ElementTree (4 points)

Le script Python de la Figure 4 agit sur le document XML de la Figure 1.

Question 1. Modifiez le script Python afin d’afficher le nom de tous les clubs.

Question 2. Modifiez le script Python afin d’afficher les clubs ayant eu plusieurs entraîneurs.

```

1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3
4 # Importation de l'API.
5 import xml.etree.ElementTree as ET
6
7 # Chemin vers le fichier xml.
8 FILE = 'entraîneurs.xml'
9
10 # Création du parser et récupération de l'arbre XML du document.
11 root = ET.parse(FILE).getroot()

```

FIGURE 4 – python.py