Examen XML

— Master d'Ingénierie Informatique —

Mars 2010, durée 2h.

Les réponses apportées aux questions doivent être justifiées avec clarté et concision. Les documents sont interdits à l'exception d'une feuille de memento (4 pages A4). L'examen est composé de quatre exercices indépendants.

▶ **Exercice 1** On considère le schéma XML suivant.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:complexType name="TypeA">
    <xsd:simpleContent>
      <xsd:extension base="xsd:integer">
        <xsd:attribute name="attr" type="xsd:string"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
  <xsd:complexType name="TypeB">
    <xsd:simpleContent>
      <xsd:restriction base="TypeA">
        <xsd:attribute name="attr" type="xsd:string" use="required"/>
      </xsd:restriction>
    </xsd:simpleContent>
  </xsd:complexType>
  <xsd:element name="elemA" type="TypeA"/>
  <xsd:element name="elemB" type="TypeB"/>
  <xsd:element name="elemC" type="TypeB" substitutionGroup="elemA"/>
  <xsd:element name="container">
    <xsd:complexType>
      <xsd:choice>
        <xsd:element ref="elemA" minOccurs="1" maxOccurs="unbounded"/>
        <xsd:element ref="elemB" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:choice>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

a) Donner un document valide pour ce schéma.

b) Donner une DTD le plus proche possible de ce schéma en signalant les différences.

c) Donner un document valide pour le nouveau schéma mais non valide pour le schéma précédent si on remplace la déclaration de l'élément `container` par la déclaration suivante.

```
<xsd:element name="container">
  <xsd:complexType>
    <xsd:choice maxOccurs="unbounded">
      <xsd:element ref="elemA" minOccurs="1"/>
      <xsd:element ref="elemB" minOccurs="1"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

d) Donner un document valide pour le schéma obtenu en ajoutant `abstract="true"` à la définition du type `TypeA` de la façon suivante.


```
<xsd:complexType name="TypeA" abstract="true">
  ...
</xsd:complexType>
```

Est-il encore possible d'utiliser l'élément `elemA` dans un document valide ? Si oui, expliquer comment.

e) Donner un document valide pour le schéma obtenu en remplaçant les déclarations des éléments `elemA` et `elemB` par les déclarations suivantes.

```
<xsd:element name="elemA" type="TypeA" abstract="true"/>
<xsd:element name="elemB" type="TypeB" abstract="true"/>
```

Est-il encore possible d'utiliser l'élément `elemA` dans un document valide ? Si oui, expliquer comment.

▶ **Exercice 2** Existe-t-il une DTD pour laquelle il n'existe pas de document valide. Si oui, donner un exemple le plus court possible.

▶ **Exercice 3** On considère des documents similaires au document suivant ou un élément `values` contient des éléments `value` avec des attributs `min` et `max` et un contenu entier.

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
<values>
  <value min="2" max="8">6</value>
  <value min="8" max="16">10</value>
</values>
```

a) Donner un schématron qui vérifie que la valeur du contenu est comprise entre la valeur de l'attribut `min` et la valeur de l'attribut `max`.

b) Modifier le schématron pour que les attributs `min` et `max` puissent être optionnels et que seuls les attributs présents soient pris en compte par le schématron.

► **Exercice 4** On considère le document suivant inspiré des données Gedcom.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<gedcom>
<indi id="I1">
  <name>Victoria /Hanover/</name>
  <titl>Queen of England</titl>
  <fams idref="F1"/>
  <famc idref="F42"/>
</indi>
<indi id="I70">
  <name>Bessiewallis /Warfield/</name>
  <fams idref="F20"/>
  <fams idref="F24"/>
  <fams idref="F25"/>
  <famc idref="F55"/>
</indi>
<indi id="I87">
  <name>William Henry Andrew /Windsor/</name>
  <titl>Prince</titl>
  <sex>M</sex>
  <famc idref="F19"/>
</indi>
</gedcom>
```

a) Donner le résultat de l'application de la feuille de style XSLT suivante sur le document précédent.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" encoding="iso-8859-1" indent="yes"/>
  <xsl:template match="/|*">
    <xsl:copy>
      <xsl:copy-of select="@*"/>
      <xsl:apply-templates/>
    </xsl:copy>
  </xsl:template>
  <xsl:template match="indi">
    <xsl:copy>
      <xsl:copy-of select="@*"/>
      <xsl:apply-templates select="*[name() != 'fams']"/>
      <xsl:if test="count(fams) = 1">
        <fams idref="{fams/@idref}" />
      </xsl:if>
      <xsl:if test="count(fams) &gt; 1">
        <fams>
          <xsl:attribute name="idrefs">
            <xsl:value-of select="fams/@idref"/>
          </xsl:attribute>
        </fams>
      </xsl:if>
    </xsl:copy>
  </xsl:template>
</xsl:stylesheet>
```

b) Donner une feuille de style qui effectue la transformation inverse. On pourra utiliser la fonction XPath `tokenize` qui découpe la chaîne passée en premier paramètre à chaque occurrence de l'expression rationnelle passée en second paramètre. Utiliser par exemple `'\s+'` comme second paramètre.