Gruppe A	PRÜFUNG AUS "SEMISTRUKTURIERTE DATEN" **181.135**		16. 06. 2011
Kennnr.	Matrikelnr.	Familienname	Vorname

Arbeitszeit: 120 Minuten. Aufgaben sind auf den Angabeblättern zu lösen; Zusatzblätter werden nicht gewertet.
Bitte tragen Sie Namen, Studienkennzahl und Matrikelnummer ein. Halten Sie Ihren Studentenausweis bereit.

Aufgabe 1: (9)

Betrachten Sie die folgende XML-Schema Datei **test.xsd**:

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

  <xsd:element name="A">
    <xsd:complexType mixed="true">
      <xsd:choice minOccurs="2" maxOccurs="3">
        <xsd:sequence>
          <xsd:element name="B" minOccurs="0" type="xsd:int"/>
        </xsd:sequence>
        <xsd:sequence>
          <xsd:element name="C" maxOccurs="3" type="myint"/>
          <xsd:element name="D" minOccurs="0" type="xsd:int"/>
        </xsd:sequence>
      </xsd:choice>
    </xsd:complexType>
  </xsd:element>
  <xsd:simpleType name="myint">
    <xsd:restriction base="xsd:int">
      <xsd:minExclusive value="1"/>
      <xsd:maxInclusive value="6"/>
    </xsd:restriction>
  </xsd:simpleType>

</xsd:schema>
```

Betrachten Sie weiters die sechs verschiedenen XML-Dateien, die unten angeführt sind.

Sie können davon ausgehen, dass alle folgenden XML-Dateien wohlgeformt sind. Es geht also lediglich darum, ihre Gültigkeit bezüglich **test.xsd** zu entscheiden.

Kreuzen Sie an, welche der folgenden XML-Dateien gültig bezüglich **test.xsd** sind.

1. `<A>abc<C>6</C>def<C>2</C><B>0</B></A>` gültig ⊗ ungültig ◯
2. `<A><C>6</C><C>2</C><C>5</C><C>4</C>abc<D>3</D></A>` gültig ⊗ ungültig ◯
3. `<A><B>10</B></A>` gültig ⊗ ungültig ◯
4. `<A><B>1</B><D>1</D>abc<C>2</C></A>` gültig ◯ ungültig ⊗
5. `<A><B>abc</B><C>1</C><D>2</D></A>` gültig ◯ ungültig ⊗
6. `<A><C>1</C><D>2</D></A>` gültig ◯ ungültig ⊗

(Regeln für Beispiele 1–3: Pro korrekter Antwort 1.5 Punkte, **pro inkorrekter Antwort -1.5 Punkte**, pro nicht beantworteter Frage 0 Punkte, für die gesamte Aufgabe mindestens 0 Punkte)

Aufgabe 2: (9)

Betrachten Sie die folgende Schema-Datei **ns.xsd**:

```
<xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.dbai.tuwien.ac.at"
    xmlns:dbai="http://www.dbai.tuwien.ac.at">

    <xsd:element name="root" type="dbai:rootType">
        <xsd:key name="k1">
            <xsd:selector xpath="./node"/>
            <xsd:field xpath="@name"/>
        </xsd:key>
    </xsd:element>

    <xsd:complexType name="rootType">
       <xsd:sequence>
          <xsd:element name="node" maxOccurs="unbounded">
             <xsd:complexType>
                <xsd:attribute name="name" type="xsd:string"/>
             </xsd:complexType>
          </xsd:element>
       </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

Kreuzen Sie an, welche Aussagen bzgl. **ns.xsd** (bzw. für ein gültiges Instanzdokument desselben) wahr bzw. falsch sind.

1. Der Präfix `xsd` ist für XML Schema fix vorgegeben und darf nicht anders heißen. wahr ◯ falsch ⊗
2. Im Instanzdokument liegen Elemente `root` und `node` im selben Namespace. wahr ◯ falsch ⊗
3. Im Instanzdokument darf man statt `dbai` auch einen anderen Präfix verwenden, sofern dieser auf den richtigen Namespace zeigt. wahr ⊗ falsch ◯
4. In der XML Schema Datei könnte man die Zeile `targetNamespace="http://www.dbai.tuwien.ac.at"` auch weglassen. wahr ◯ falsch ⊗
5. Die Schlüsseldefinition ist ungültig, da kein Fremdschüssel definiert wird. wahr ◯ falsch ⊗
6. Der Pfad für den Schlüssel ist nicht korrekt, da der Präfix fehlt. wahr ◯ falsch ⊗

Aufgabe 3: (9)

Kreuzen Sie an, ob die folgenden Aussagen wahr oder falsch sind.

1. Um die Wohlgeformtheit eines XML-Dokuments zu überprüfen wird eine DTD oder ein XML-Schema benötigt. wahr ◯ falsch ⊗
2. Die DTD Elementdeklaration `(A+|B+|C+)` ist gleichwertig mit `(A|B|C)+`. wahr ◯ falsch ⊗
3. Die Ordnung der Knoten im Resultat eines XPath-Ausdrucks ist immer in Document Order. wahr ◯ falsch ⊗
4. Ein XSLT-Stylesheet muss ein wohlgeformtes XML-Dokument sein. wahr ⊗ falsch ◯
5. SAX verwendet die DOM API um ein XML Dokument zu lesen. wahr ◯ falsch ⊗
6. SAX: Textevents können auch direkt hintereinander auftreten. wahr ⊗ falsch ◯

Die folgenden Aufgaben 4 – 7 beziehen sich auf das XML-Dokument textref.xml, das Sie auf der letzten Seite dieser Prüfungsangabe finden.

Aufgabe 4: (12)

Vervollständigen Sie die DTD **textref.dtd**, sodass XML-Dokumente in der Gestalt von **textref.xml** (siehe Anhang) bezüglich dieser DTD gültig sind. Berücksichtigen Sie beim Erstellen der DTD folgende Punkte:

- `document` beinhaltet genau ein `main` Element, maximal ein `comment` Element und beliebig viele `part` Elemente. `part` Elemente dürfen nur am Ende vorkommen. Das `comment` Element ist optional, darf aber, sofern es vorkommt, nur direkt vor oder nach dem `main` Element stehen.
- Der Inhalt des `main` Elements soll gemischt sein, es dürfen als Subelemente `author` und `ref` Elemente auftreten.
- `part` Elemente haben gemischten Inhalt mit Subelementen `ref` und
  - ein Attribut `id` das nicht fehlen darf und einen eindeutigen Attributwert beinhalten soll.
- `ref` Elemente besitzen keinen Inhalt, lediglich
  - ein Attribut `to` welches nicht fehlen darf und auf die `id` eines `part` Elements referenziert.
- `author` Elemente und das `comment` Element sollen Textinhalte haben.

Datei **textref.dtd**:

```
<!ELEMENT document (((comment, main) | (main, comment?)),  part*)>
<!ELEMENT main (#PCDATA | author | ref)*>
<!ELEMENT part (#PCDATA | ref)*>
<!ATTLIST part id ID #REQUIRED>
<!ELEMENT ref EMPTY>
<!ATTLIST ref to IDREF #REQUIRED>
<!ELEMENT author (#PCDATA)>
<!ELEMENT comment (#PCDATA)>
```

Aufgabe 5: (10)

Betrachten Sie die folgenden XPath-Abfragen angewandt auf das Dokument **textref.xml** (siehe Anhang).

- Falls als Ergebnis eine Knotenmenge selektiert wird, geben Sie als Ausgabe die Werte der `id` Attribute an.
- Falls der angegebene XPath Ausdruck keine Knoten selektiert, notieren Sie im entsprechenden Feld "leere Ausgabe".
- Falls als Ergebnis eine Zahl selektiert wird (count), geben Sie diese Zahl an.

Betrachten Sie dazu folgendes Beispiel:

```
//part
```

p1 p2 p3 p4

Geben Sie nun die entsprechende Ausgaben der folgenden XPath-Abfragen an.

```
count(//ref[@to='p1'])
```

2

```
//part[*]
```

p2 p4

```
/document/*[2]
```

p1

```
//part[@id = //ref/@to]
```

p1 p2 p3 p4

```
//part[@id = ref/@to]
```

p4

Aufgabe 6: (9)

Erstellen Sie ein XSLT-Stylesheet **textref.xsl**, das angewandt auf Dokumente der Gestalt **textref.xml** folgende Ausgabe liefert:

- Zurückgegeben werden soll der Inhalt des `main` Elements.
- Statt der `ref` Elemente soll der Inhalt der entsprechenden `part` Elemente zurückgegeben werden. Das gilt sowohl für direkt in `main` enthaltene Referenzen, als auch für in `part` enthaltene Referenzen.

Die Ausgabe soll reiner Text sein (d.h. kein XML Markup enthalten).

Betrachten Sie dazu folgende Ausgabe, die ihr XSLT-Stylesheet **textref.xsl** angewandt auf **textref.xml** (siehe Anhang) produzieren soll:

```
Das ist ein Text, der ein interessantes, aber komplexes Beispiel veranschaulicht.
```

Vervollständigen Sie hier das XSLT-Stylesheet **textref.xsl**. Kontrollstrukturen wie z.B. `xsl:for-each` sind für die Lösung grundsätzlich erlaubt aber nicht erforderlich (ausreichend zur Lösung des Beispiels sind wenige Templates mit jeweils relativ kurzem Inhalt). Sie brauchen sich nicht um Whitespaces etc. zu kümmern.

Datei **textref.xsl**:

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

<xsl:template match="/">
    <xsl:apply-templates select="//main/node()"/>
</xsl:template>

<xsl:template match="ref">
    <xsl:apply-templates select="//part[@id=current()/@to]"/>
</xsl:template>

</xsl:stylesheet>
```

Aufgabe 7: (9)

Betrachten Sie folgende-XQuery Abfrage **textref.xq**:

```
<statistics>{

  for $a in doc('textref.xml')/document/*
  let $count := count($a//*)
  return element {concat('c', $count)}
      {attribute {'t'}
          {$a/@id}}

}</statistics>
```

Geben Sie nun die Ausgabe von **textref.xq** angewandt auf **textref.xml** an.

Die exakte Behandlung von Whitespaces ist für diese Beispiel nicht relevant.

```
<statistics>

   <c3 t=""/>
   <c0 t="p1"/>
   <c1 t="p2"/>
   <c0 t="p3"/>
   <c1 t="p4"/>

</statistics>
```

Aufgabe 8: (8)

Vervollständigen Sie die Methode `editRef`, die angewandt auf das DOM Element `ref` (ein `ref` Element aus einem Dokument der Form **textref.xml**) folgende Änderung durchführt: Das `ref` Element der Form

```
<ref to='px'/>
```

wird ersetzt durch ein Element der Form

```
<include part='px'/>
```

Die Methode `editRef` erhält weiters den Elternknoten des `ref` Elements. Dieser ist erforderlich, um die Ersetzung durchführen zu können.

Um Fehlerbehandlung müssen Sie sich nicht kümmern.

```
public static void editRef(Element ref, Element parent) {
  String to = ref.getAttribute("to");
  Element include = ref.getOwnerDocument().createElement("include");
  include.setAttribute("part", to);

  parent.replaceChild(include, ref);
}
```

Sie können diese Seite abtrennen!

Datei **textref.xml**:

```
<document>

    <main>
        Das ist <ref to="p1"/> Text,
        der <ref to="p2"/>, aber
        <ref to="p3"/> Beispiel veranschaulicht.
    </main>

    <part id="p1">ein</part>
    <part id="p2"><ref to="p1"/> interessantes</part>
    <part id="p3">komplexes</part>
    <part id="p4">eigenartiges <ref to="p4"/></part>

</document>
```

Gesamtpunkte: 75