Gruppe A	PRÜFUNG AUS "SEMISTRUKTURIERTE DATEN" **181.135**		21. 01. 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="X">
    <xsd:complexType mixed="true">
      <xsd:choice minOccurs="1" maxOccurs="2">
        <xsd:sequence>
          <xsd:element name="Y" maxOccurs="2" type="xsd:string"/>
        </xsd:sequence>
        <xsd:sequence>
          <xsd:element name="Z" minOccurs="0" type="xsd:integer"/>
        </xsd:sequence>
     </xsd:choice>
    </xsd:complexType>
  </xsd:element>
</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. `<X><Y><Z>123</Z></Y></X>` gültig ◯ ungültig ◯
2. `<X><Y>123</Y></X>` gültig ◯ ungültig ◯
3. `<X><Y>123</Y><Y>456</Y><Y>789</Y></X>` gültig ◯ ungültig ◯
4. `<X><Z>123</Z><Z>456</Z><Z>789</Z></X>` gültig ◯ ungültig ◯
5. `<X></X>` gültig ◯ ungültig ◯
6. `<X>123</X>` gültig ◯ ungültig ◯

(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: (6)

Betrachten Sie die folgende XML- Datei **ns.xml**:

```
<A xmlns:ns="www.NS1.at">
  <ns:B xmlns="www.NS2.at">
    <D>abc</D>
  </ns:B>
  <ns:C xmlns:ns2="www.NS2.at">
    <ns2:E><F/></ns2:E>
  </ns:C>
</A>
```

Kreuzen Sie an, ob die folgenden Aussagen für die Datei **ns.xml** wahr oder falsch sind.

1. Elemente `A` und `B` liegen im selben Namespace. wahr ◯ falsch ◯
2. Elemente `B` und `C` liegen im selben Namespace. wahr ◯ falsch ◯
3. Elemente `D` und `E` liegen im selben Namespace. wahr ◯ falsch ◯
4. Elemente `E` und `F` liegen im selben Namespace. wahr ◯ falsch ◯

(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 3: (12)

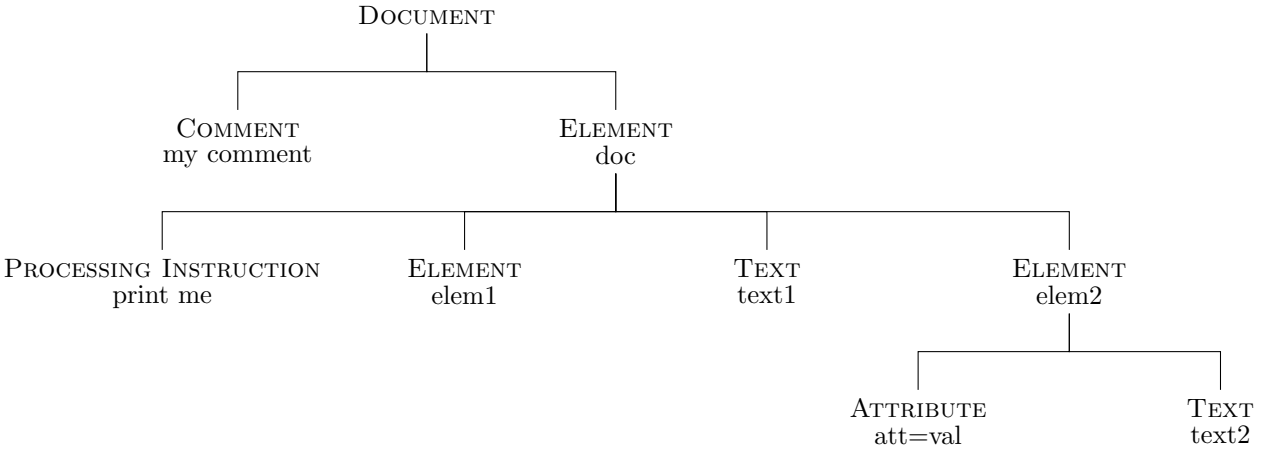
Kreuzen Sie an, ob die folgenden Aussagen wahr oder falsch sind.

1. Eine XML Schema Definition muss ein wohlgeformtes XML-Dokument sein. wahr ◯ falsch ◯
2. Der Speicherbedarf eines DOM-Parsers ist unabhängig von der Größe des geparsten XML-Files. wahr ◯ falsch ◯
3. Die Ordnung der Knoten im Resultat eines XPath-Ausdrucks ist immer in Document-Order. wahr ◯ falsch ◯
4. Um die Wohlgeformtheit eines XML-Dokuments zu überprüfen wird eine DTD oder ein XML Schema benötigt. wahr ◯ falsch ◯
5. Die DTD Elementdeklaration `(A?|B?|C?)` ist gleichwertig mit `(A|B|C)?`. wahr ◯ falsch ◯
6. SAX Textevents treten niemals direkt hintereinander auf. wahr ◯ falsch ◯
7. Wenn in XSLT für einen Knoten kein Template angegeben wird, dann wird die Verarbeitung abgebrochen, falls dieser in der Eingabe auftritt. wahr ◯ falsch ◯
8. Das Ergebnis eines XSLT-Stylesheets ist immer ein wohlgeformtes XML Dokument. wahr ◯ falsch ◯

(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 4: (7)

Geben Sie das XML-Dokument an, das dem folgenden DOM-Baum entspricht. Die XML-Deklaration (<?xml...?>) brauchen Sie nicht zu berücksichtigen.



Die folgenden Aufgaben 5 – 8 beziehen sich auf das XML-Dokument gutachten.xml, das Sie auf der letzten Seite dieser Prüfungsangabe finden.

Aufgabe 5: (12)

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

- Das Wurzelelement `gutachten` enthält beliebig viele Elemente `person`, gefolgt von beliebig vielen Elementen `artikel`. Stellen Sie sicher, dass zumindest ein `person` Element vorkommt; für `artikel` ist das nicht notwendig.
- `person` ist ein leeres Element mit den Attributen `name` und `nr`; `nr` soll innerhalb des Dokuments eindeutig sein.
- Ein `artikel` Element hat ein Attribut `titel` sowie die folgenden Kindelemente:
  - `verfasst-von` soll zumindest einmal, kann aber beliebig auf, auftreten;
  - danach folgen zumindest ein, aber maximal drei, Elemente `bewertung`.
    - Der Inhalt eines `bewertung` Elements soll Text sein.
    - `bewertung` hat zwei Attribute, wobei der Attributwert des `von`- Attributs auf eine Personennummer verweisen soll; für das Attribut `beurteilung` sollen nur Werte zwischen 1 und 5 (wie Schulnoten) erlaubt sein.
- Alle Attribute sollen verpflichtend sein. Beachten Sie auch die Beziehung zwischen Attributen und bilden Sie diese in der DTD entsprechend ab!

Datei **gutachten.dtd**:

Aufgabe 6: (10)

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

- Falls als Ergebnis eine Knotenmenge selektiert wird, verwenden Sie als Ausgabe
  - bei `person` Elementen den Wert des `nr` Attributes
  - bei `artikel` Elementen den Wert des `titel` Attributes
- Falls der angegebene XPath Ausdruck die leere Knotenmenge selektiert, notieren Sie im Feld "leere Ausgabe".
- Falls als Ergebnis eine Zahl selektiert wird (`count`), geben Sie diese Zahl an.

Betrachten Sie dazu folgendes Beispiel:

```
//person | //artikel
```

MM SS PF FF AD Artikel 1 Artikel 2

Geben Sie nun für die folgenden XPath-Anfragen die entsprechenden Ausgaben an.

```
count(//bewertung[@von='SS'])
```


```
//artikel[count(bewertung) > 2]
```


```
//artikel[verfasst-von=bewertung/@von]
```


```
//person[@nr=//@von]
```


```
//person[not(@nr=//verfasst-von)]
```


Aufgabe 7: (10)

Betrachten Sie das folgende XSLT-Stylesheet **gutachten.xsl**.

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
   <xsl:template match="/">
      <autoren>
         <xsl:apply-templates select="//person">
            <xsl:sort select="@name" order="descending"/>
         </xsl:apply-templates>
      </autoren>
   </xsl:template>

   <xsl:template match="person">
      <autor name="{@name}">
         <xsl:apply-templates select="//artikel[verfasst-von=current()/@nr]"/>
      </autor>
   </xsl:template>

   <xsl:template match="artikel">
      <xsl:if test="count(verfasst-von) &lt; 3">
         <artikel titel="{@titel}"/>
      </xsl:if>
   </xsl:template>
</xsl:stylesheet>
```

Vervollständigen Sie hier die XQuery-Anfrage **gutachten.xq**. Sie soll für alle XML-Dokumente der Gestalt **gutachten.xml** (siehe Anhang) die gleichen Ergebnisse liefern wie **gutachten.xsl**. Sie brauchen sich nicht um Whitespaces etc. zu kümmern.

Nehmen Sie an, dass der Kontextknoten der nachfolgenden Query der Dokumentknoten von einem Dokument der Gestalt **gutachten.xml** ist! (d.h. `//person` selektiert alle `person` Elemente in diesem Dokument, etc.).

Datei **gutachten.xq**:

Aufgabe 8: (9)

Nehmen Sie an, dass folgender `XMLFilter` auf das XML-Dokument **gutachten.xml** (siehe Anhang) ausgeführt wird. Schreiben Sie auf, welche SAX-Events die Applikation weitergereicht bekommt.

Eventnamen dürfen Sie abkürzen.

```
public class SSDFilter extends XMLFilterImpl {
   public void startElement(String uri, String localName, String qName, Attributes atts)
         throws SAXException {
      if ("gutachten".equals(localName)) {
         super.startElement(uri, localName, qName, atts);
      }

      if ("artikel".equals(localName)) {
         String titel = atts.getValue("titel");

         super.startElement(uri, localName, qName, null);
         super.characters(titel.toCharArray(), 0, titel.length());
         super.endElement(uri, localName, qName);
      }
   }

   public void endElement(String uri, String localName, String qName) throws SAXException {
      if ("gutachten".equals(localName)) {
         super.endElement(uri, localName, qName);
      }
   }

   public void characters(char []ch, int start, int length) throws SAXException {
   }
}
```

Eventname	Elementname, Attibute, Textinhalt etc...

Sie können diese Seite abtrennen!

Datei **gutachten.xml**:

```
<gutachten>
   <person name="Max Mustermann" nr="MM"/>
   <person name="Sabine Standard" nr="SS"/>
   <person name="Paul Faul" nr="PF"/>
   <person name="Frieda Fleissig" nr="FF"/>
   <person name="Advocatus Diaboli" nr="AD"/>

   <artikel titel="Artikel 1">
      <verfasst-von>MM</verfasst-von>
      <verfasst-von>PF</verfasst-von>
      <verfasst-von>FF</verfasst-von>

      <bewertung von="SS" beurteilung="1">
         Super toller Artikel!
      </bewertung>
   </artikel>

   <artikel titel="Artikel 2">
      <verfasst-von>SS</verfasst-von>
      <verfasst-von>PF</verfasst-von>

      <bewertung von="MM" beurteilung="1"/>
      <bewertung von="FF" beurteilung="2">
         Ok!
      </bewertung>
      <bewertung von="AD" beurteilung="5">
         Artikel kann so nicht akzeptiert werden.
      </bewertung>
   </artikel>
</gutachten>
```

Gesamtpunkte: 75