Gruppe A

Bitte tragen Sie **sofort** und **leserlich** Namen, Studienkennzahl und Matrikelnummer ein und legen Sie Ihren Studentenausweis bereit.

PRÜFUNG AUS "SEMISTRUKTURIERTE DATEN" **181.135**			17. 10. 2008
Kennnr.	Matrikelnr.	Familienname	Vorname

Arbeitszeit: 120 Minuten. Aufgaben sind auf den Angabeblättern zu lösen; Zusatzblätter werden nicht gewertet.

Aufgabe 1: (9)

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

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="A">
  <xsd:complexType>
    <xsd:choice minOccurs="0" maxOccurs="3">
       <xsd:element name="B" type="AType" minOccurs="1" maxOccurs="2"/>
       <xsd:element name="C" type="AType" minOccurs="0" maxOccurs="1"/>
    </xsd:choice>
  </xsd:complexType>
 </xsd:element>
 <xsd:complexType name="AType">
   <xsd:attribute name="D" type="xsd:string"/>
   <xsd:attribute name="E" type="xsd:string"/>
 </xsd:complexType>
</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>test</A>` gültig ◯ ungültig ⊗
2. `<A><C D="att" E="att2"><B D="att3" E="att4"/></C></A>` gültig ◯ ungültig ⊗
3. `<A><B E="att" D="att2"/></A>` gültig ⊗ ungültig ◯
4. `<A><C/><B/><B/><C/></A>` gültig ⊗ ungültig ◯
5. `<A><B/><C/><C/><B/></A>` gültig ◯ ungültig ⊗
6. `<A><B D="att">E</B><C></C></A>` 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: (9)

Geben Sie eine DTD **docu.dtd** an, sodass das XML-Dokument **docu.xml** (siehe Anhang) bezüglich dieser DTD gültig ist. Berücksichtigen Sie beim Erstellen der DTD folgende Punkte:

- Die Verknüpfung mit einer XML Schema-Definition ist optional; alle anderen Attribute, die Sie verwenden, sollen verpflichtend sein.
- Das "dokument"-Element beinhaltet eine beliebige Abfolge von "newline" und "text" Elementen. Stellen Sie aber sicher, dass zumindest ein "text"-Element vorkommt. Beachten Sie, dass "newline" ein leeres Element mit Attribut ist.
- Das "text"-Element hat gemischten Inhalt und kann beliebig viele (auch keine) Subelemente "b", "i" beinhalten. Die Reihenfolge des Auftretens der einzelnen Subelemente soll beliebig sein. "b" und "i" sollen einfache Elemente mit Stringinhalt sein (d.h. diese Elemente dürfen nicht verschachtelt auftreten).

Datei **docu.dtd**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT format:dokument
    (format:newline*,format:text,(format:newline|format:text)*)>
<!ATTLIST format:dokument
    xmlns:format CDATA #REQUIRED
    xmlns:xsi CDATA #REQUIRED
    xsi:schemaLocation CDATA #IMPLIED>
<!ELEMENT format:newline EMPTY>
<!ATTLIST format:newline
    abstand CDATA #REQUIRED>
<!ELEMENT format:text
    (#PCDATA | i | b)*>
<!ELEMENT i (#PCDATA)>
<!ELEMENT b (#PCDATA)>
```

Aufgabe 3: (12)

Geben Sie nun eine XML Schema Definition **docu.xsd** an, sodass das XML-Dokument **docu.xml** (siehe Anhang) bezüglich dieses Schemas gültig ist.

- Es gelten dieselben Hinweise wie für Beispiel 2. Sie können sich (müssen aber nicht) neue Typen explizit definieren.
- Außerdem soll sichergestellt sein, dass das "abstand"-Attribut des "newline"-Elements nur positive Zahlen beinhaltet.

Datei **docu.xsd**:

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

   <xsd:element name="dokument">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="newline" form="qualified"
                         minOccurs="0" maxOccurs="unbounded" type="f:newlineType"/>
            <xsd:sequence maxOccurs="unbounded">
               <xsd:element name="text" form="qualified">
                  <xsd:complexType mixed="true">
                     <xsd:choice minOccurs="0" maxOccurs="unbounded">
                        <xsd:element name="b" type="xsd:string"/>
                        <xsd:element name="i" type="xsd:string"/>
                     </xsd:choice>
                  </xsd:complexType>
               </xsd:element>
               <xsd:element name="newline" form="qualified"
                            minOccurs="0" maxOccurs="unbounded" type="f:newlineType"/>
            </xsd:sequence>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
   <xsd:complexType name="newlineType">
      <xsd:attribute name= "abstand" type="xsd:positiveInteger" use="required"/>
   </xsd:complexType>
</xsd:schema>
```

Aufgabe 4: (9)

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

```
<?xml version="1.0" encoding="UTF-8"?>
<liste>
   <element>eins</element>
   <element>zwei</element>
</liste>
```

Betrachten Sie weiters dieses leere XSLT-Stylesheet **query.xsl**:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="text" indent="yes" encoding="UTF-8" version="1.0" />

<!-- Platzhalter -->

</xsl:stylesheet>
```

Im folgenden finden Sie 6 Möglichkeiten wie der Kommentar `<!-- Platzhalter -->` ersetzt werden kann. Sie können davon ausgehen, dass das XSLT-Stylesheet syntaktisch korrekt bleibt, d.h. konzentrieren Sie sich auf die Funktionalität.

Kreuzen Sie an für welche der Ersetzungen des Kommentars `<!-- Platzhalter -->` das Stylesheet angewandt auf **simple.xml** folgenden Output ergibt:

```
eins
zwei
```

(die exakte Behandlung von Whitespaces ist nicht relevant):

1.
```
<xsl:template match="element">
    <xsl:value-of select="."/>
</xsl:template>
```
ja ⊗ nein ◯

2. `<xsl:template match="text()"/>` ja ◯ nein ⊗

3.
```
<xsl:template match="/">
        <xsl:value-of select="liste/element"/>
</xsl:template>
```
ja ◯ nein ⊗

4.
```
<xsl:template match="liste">
        <xsl:apply-templates/>
</xsl:template>
```
ja ⊗ nein ◯

5.
```
<xsl:template match="/">
        <xsl:apply-templates select="liste/element[1]"/>
        <xsl:apply-templates select="liste/element[2]"/>
</xsl:template>
```
ja ⊗ nein ◯

6. `<!-- Es ist kein weiterer Code notwendig. -->` ja ⊗ nein ◯

(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 5: (7)

Geben Sie das XML Dokument an, dass folgenden SAX Events entspricht. Die XML Deklaration (<?xml...?>) brauchen Sie nicht zu berücksichtigen.

startDocument	
startElement	dok
startElement	wort [lang: de]
characterEvent	tantieme
endElement	wort
startElement	wort [lang: en]
characterEvent	royality
endElement	wort
endElement	dok
endDocument	

```
<dok><wort lang="de">tantieme</wort><wort lang="en">royality</wort></dok>
```

Aufgabe 6: (9)

Vervollständigen Sie die folgende Java Methode sodass die Anzahl der Attribute eines übergebenen XML Dokuments berechnet wird. Die Methode zählt also die Anzahl der Attribute im übergebenen Dokument und gibt die Summe als Rückgabewert zurück.

Tipp: Navigieren Sie **rekursiv** mithilfe der Methoden aus den DOM-Folien durch den Baum. Die Anzahl der Attribute ergibt sich aus der Anzahl der Attribute des aktuellen Knoten plus die Anzahl der Attribute der Sub-Knoten.

```
int sumNumberOfAttributes(Node node) throws Exception {

  int anzahl = node.getAttributes().getLength();
  NodeList nl = node.getChildNodes();
  for(int i=0; i < nl.getLength(); ++i) {
    Node n = nl.item(i);
    anzahl = anzahl + sumNumberOfAttributes(n);
  }
  return anzahl;

}
```

Aufgabe 7: (8)
Schreiben Sie XPath-Anfragen um für XML-Dokumente wie **docu.xml** (siehe Anhang) folgende Informationen zu selektieren:

1. Alle "newline" Elemente die ein "abstand" Attribut mit einem Wert größer als 1 haben.
   ```
   //format:newline[@abstand > 1]
   ```

2. Selektieren Sie jedes zweite "text" Element. Dh. das Erste, das Dritte etc. Tipp: Modulo!
   ```
   /format:dokument/format:text[position() mod 2 = 1]
   ```

3. Das letzte "text" Element in dem mindestens ein "i" Element als Sub-Element vorkommt.
   ```
   //format:text[i][position() = last()]
   ```

4. Den Durchschnitt der Werte der "abstand" Attribute.
   ```
   sum(//node()/@abstand) div count(//node()/@abstand)
   ```

Aufgabe 8: (12)
Kreuzen Sie an, ob die folgenden Aussagen wahr oder falsch sind.

1. Folgender Ausdruck liefert dann ein nicht-leeres Ergebnis wenn mindestens ein "subknoten"-Element von "knoten" den Inhalt "drei" hat:
   `//knoten[subknoten = "drei"]` wahr ⊗ falsch ◯
2. Attribute werden bei einem SAX Parser nicht als eigene Events getriggert. wahr ⊗ falsch ◯
3. Bei einem "Pull-Parser" kontrolliert die Applikation den Programmablauf. wahr ⊗ falsch ◯
4. Folgende zwei Ausdrücke sind äquivalent:
   `//element[2][@attribut >= 2]`
   `//element[@attribut >= 2][2]` wahr ◯ falsch ⊗
5. Der Speicherbedarf bei einem DOM Parser ist nicht abhängig von der Größe des XML Dokuments. wahr ◯ falsch ⊗
6. SAX verwendet die DOM API um ein XML Dokument zu lesen. wahr ◯ falsch ⊗
7. Namespace-Attribute wie z.B. `xmlns` müssen in DTDs nicht extra spezifiziert werden wahr ◯ falsch ⊗
8. Der XPath-Ausdruck `./a[@b=c]` ist die Kurzschreibweise des XPath-Ausdrucks
   `self::node()/child::a/attribute::b[. = c]` 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)

XML-Dokument **docu.xml** (Sie können diese Seite abtrennen!)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE format:dokument SYSTEM "docu.dtd">
<format:dokument
    xmlns:format= "http://www.dbai.tuwien.ac.at/docu"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.dbai.tuwien.ac.at/docu/docu.xsd">
    <format:text>
        Dies ist ein Beispieltext fuer die <b>SSD-Pruefung</b>.
        <i>Lesen Sie die naechsten beiden Absaetze genau durch</i>.
    </format:text>
    <format:newline abstand="4"/>
    <format:text>
        <i>Innerhalb eines i-tags darf kein weiterer tag vorkommen;
        also weder ein i-tag noch ein b-tag.</i>
    </format:text>
    <format:text>
        <b>Ebenso darf innerhalb eines b-tags darf kein weiterer tag
        vorkommen; also wiederum weder ein i-tag noch ein b-tag.</b>
    </format:text>
    <format:newline abstand="1"/>
    <format:newline abstand="1"/>
</format:dokument>
```

Gesamtpunkte: 75