Klausur zur Vorlesung „Einführung in XML“

Nachname: Vorname:

Matr.Nr.: Studiengang:

Bearbeiten Sie alle Aufgaben! Bei Ankreuzaufgaben können mehrere Antworten richtig sein. Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 120 Minuten.

Aufgabe	Punkte max.	Punkte erreicht
1	4	
2	2	
3	10	
4	10	
5	8	
6	3	
7	4	
8	4	
9	6	
Summe	**51**	

Aufgabe 1:

Diese Klausur ist ein reines Glücksspiel. Das folgende XML-Dokument beschreibt vereinfacht einen Teil des Spielstands in einem Skatspiel. Es genügt zu wissen: Skat ist ein Kartenspiel für drei Spieler (Hände), das Blatt hat 32 Karten mit *Ass*, *Zehn*, *König*, *Dame*, *Bube*, *Neun*, *Acht*, *Sieben* in den Farben *Kreuz* (♣), *Pik* (♠), *Herz* (♥) und *Karo* (♦). Es werden 30 Karten ausgeteilt, 2 Karten bleiben verdeckt liegen und bilden den Skat. Die *Augen* geben den Wert einer Karte an.

Markieren Sie die Fehler im folgenden, nicht wohlgeformten Dokument!

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="Skat.xsl" ?>
<!DOCTYPE Spiel SYSTEM "skat.dtd">
  <Spiel>
    <Hand Position='vor'>
      <Karte Farbe="Pik" Augen=10>Zehn</Karte>
      <Karte Farbe="Pik" Augen=2>Bube</Karte>
      <Karte Farbe="Karo" Augen=11>Ass</Karte>
    </HAND>
    <Hand Position='mittel'>
      <Karte Farbe="Kreuz" Augen="11">Ass</Karte>
      <Karte Farbe="Herz" Augen="0">Sieben</Karte>
      <Karte Farbe="Karo" Augen="3">Dame</Karte>
    </Hand>
    <Hand Position='hinter'>
      <Karte Farbe="Kreuz" Augen="10">Zehn</Karte>
      <Karte Farbe="Pik" Augen="11">Ass</Karte>
      <Karte Farbe="Karo" Augen="4">Koenig</Karte>
    </Hand>
    <Skat>
      <Karte Farbe = "Herz" Augen = "10">Zehn</Karte>
      <Karte Farbe = "Pik" Augen = "0">Neun</Karte>
    </Skat>
  </Spiel>
```

Aufgabe 2:

Im XML-Dokument aus Aufgabe 1 ist das **Wurzelelement**

( ) **`<?xml ...>`**.

( ) **`<Spiel>`**.

( ) ein nicht sichtbarer Knoten oberhalb von **`<Spiel>`** und **`<?xml ...>`**.

( ) nicht definiert, weil keine DTD angegeben wurde.

Aufgabe 3:

Eine DTD für die Skatspiele aus Aufgabe 1 sieht wie folgt aus.

```
<!-- DTD fuer Skat -->
<!ELEMENT Spiel (Hand, Hand, Hand, Skat)>
<!ATTLIST Spiel Trumpf CDATA #IMPLIED>
<!ELEMENT Hand (Karte)*>
<!ATTLIST Hand Position ( vor | mittel | hinter ) #REQUIRED>
<!ELEMENT Skat (Karte, Karte)>
<!ELEMENT Karte (#PCDATA)>
<!ATTLIST Karte Farbe ( Kreuz | Pik | Herz | Karo ) #REQUIRED>
<!ATTLIST Karte Augen CDATA #REQUIRED>
```

(a) Wie viele Karten hat eine **`Hand`** mindestens?

(b) Welche **Attribute** sind optional?

(c) Gibt es eine elegantere Methode in einer DTD, um zu erreichen, daß es im Element **`<Spiel>`** genau drei Hände gibt? Wenn ja, wie lautet die Angabe?

(d) Gibt es ein Element, für das **gemischter Inhalt** erlaubt ist? Wenn ja, für welches?

(e) Das **`Augen`**-Attribut aus **`Karte`** darf nicht weggelassen werden und sollte eigentlich nur die Integer-Werte 11, 10, 4, 3, 2 oder 0 annehmen. Ändern Sie die Attributdefinition, damit nur diese Werte als gültig (valid) beim Parsen akzeptiert werden?

Aufgabe 4:

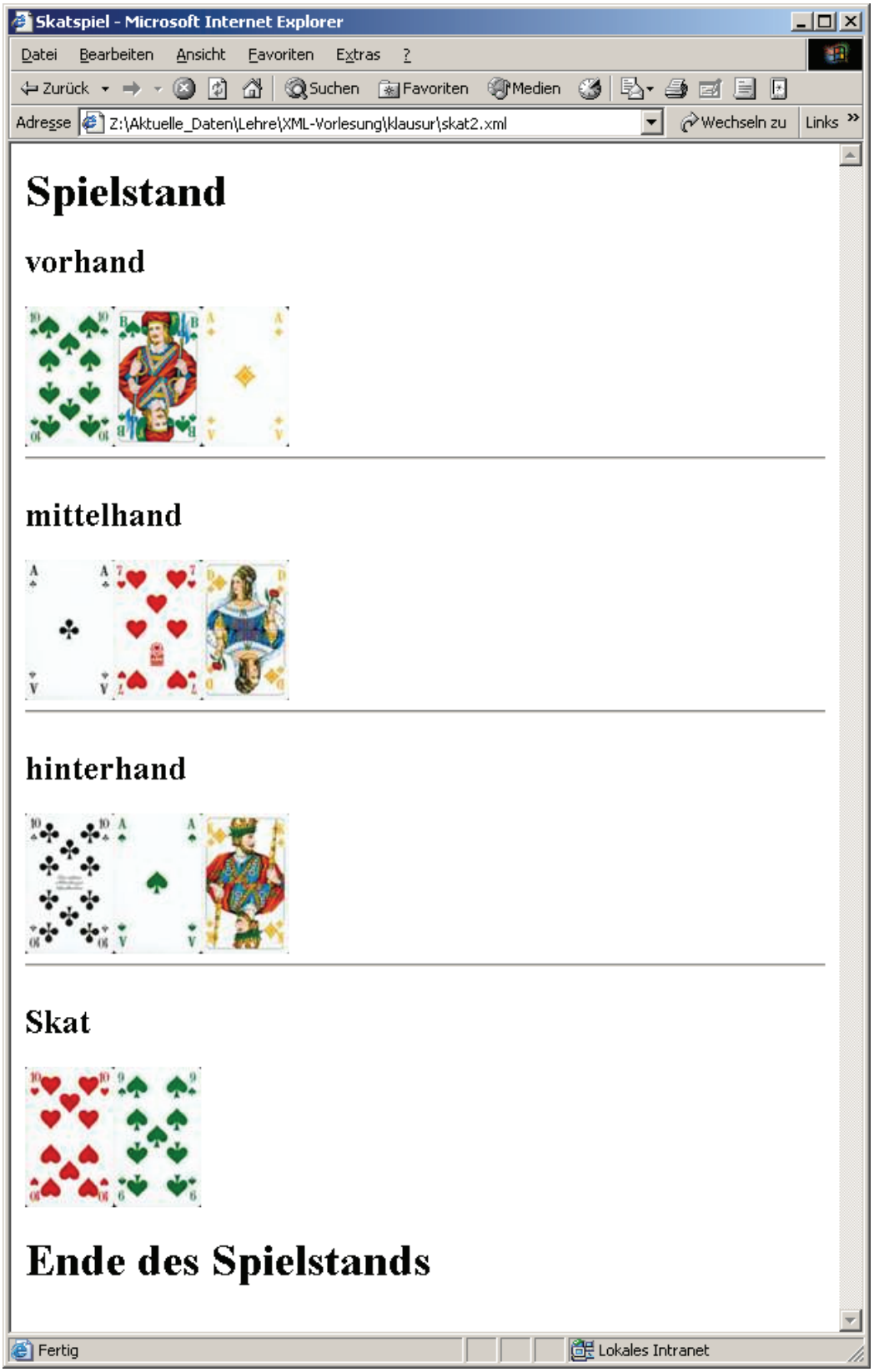
Wie gesagt: reines Glücksspiel. Geben Sie ein gültiges XML-Dokument (einen Lottoschein) zu folgender XML Schema Definition an. Der Lottoschein soll das **Art**-Attribut **"Normal-schein"** haben.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:annotation>
  <xsd:documentation xml:lang="de-DE">Schema fuer einen
  vereinfachten Lottoschein.</xsd:documentation>
</xsd:annotation>
<xsd:element name="Lottoschein" type="LScheint" />
<xsd:complexType name="LScheint">
  <xsd:sequence>
    <xsd:element name="Felder" type="Feldert" />
    <xsd:element name="Los-Nummer" type="xsd:integer" />
    <xsd:element name="ersterZiehungstag" type="xsd:string" />
  </xsd:sequence>
  <xsd:attribute name="Art" type="xsd:string"
                 use="required" />
  <xsd:attribute name="LaufzeitWochen" type="LZt"
                 use="optional" default="1" />
  <xsd:attribute name="Ziehungstage" type="xsd:string"
                 use="optional" default="Sa" />
</xsd:complexType>
<xsd:complexType name="Feldert">
  <xsd:sequence>
    <xsd:element name="Lottofeld" type="LFt"
      minOccurs="1" maxOccurs="12" />
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="LFt">
  <xsd:sequence>
    <xsd:element name="Z" type="lottoZahl"
      minOccurs="6" maxOccurs="6" />
  </xsd:sequence>
  <xsd:attribute name="Nr" type="xsd:integer" use="required" />
</xsd:complexType>
<xsd:simpleType name="lottoZahl">
  <xsd:restriction base="xsd:integer">
```

```
    <xsd:minExclusive value="0" />
    <xsd:maxInclusive value="49" />
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="LZt">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="1"/>
    <xsd:enumeration value="2"/>
    <xsd:enumeration value="3"/>
    <xsd:enumeration value="4"/>
    <xsd:enumeration value="Dauer"/>
  </xsd:restriction>
</xsd:simpleType>
</xsd:schema>
```

Aufgabe 5:

Die nebenstehende Abbildung zeigt die HTML-Ausgabe des XML-Skatblatts aus Aufgabe 1. Die Ausgabe wurde mit dem Stylesheet unten erzeugt. Vervollständigen Sie das Stylesheet!



```
<?xml version='1.0' ?>
<xsl:stylesheet version='1.0'
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
    <head>
        <title>Skatspiel</title>
    </head>
    <body>
        <h1>_____________________________________________</h1>
        <xsl:apply-templates />
        <h1>Ende des Spielstands</h1>
    </body>
  </html>
</xsl:template>

<xsl:template match="_____________________________">

  <P><h2><xsl:value-of select="_________________________"/>hand </h2>
  <xsl:apply-templates />
  <HR/></P>
</xsl:template>

<xsl:template match="Karte">
  <img src="{concat(@Farbe,text(),'.jpg')}"/>
</xsl:template>

<xsl:template match="Skat">
    <P><h2>Skat</h2></P>
    ____________________________________________________________
</xsl:template>
</xsl:stylesheet>
```

Aufgabe 6:

Wie müssen die JPG-Dateien heißen, damit das Stylesheet oben die passenden Bilder anzeigt? Nennen Sie die Dateinamen der drei JPG-Dateien der Vorhand im Bild oben.

_____ _____ _____

Aufgabe 7:

Die folgende XQuery soll auf dem Skatblatt (skat2.xml) aus Aufgabe 1 laufen, wobei das Dokument dann natürlich wohlgeformt sein muß. Welche Ausgabe wird geliefert?

```
<Augensummen>
{
  for $hand in fn:doc("skat2.xml")/Spiel/Hand
  return
    <Hand Position="{$hand/@Position}">
    {
      fn:sum($hand/Karte/@Augen)
    }
    </Hand>
}
</Augensummen>
```

Aufgabe 8:

Gegeben sei die folgende Datenbanktabelle SPIELE mit drei Spielen auf zwei Lottoscheinen.

SCHEINNR	SPIELNR	Z1	Z2	Z3	Z4	Z5	Z6
4711	1	5	7	12	20	31	42
4711	2	4	11	22	23	27	44
9362	1	9	10	30	41	42	49

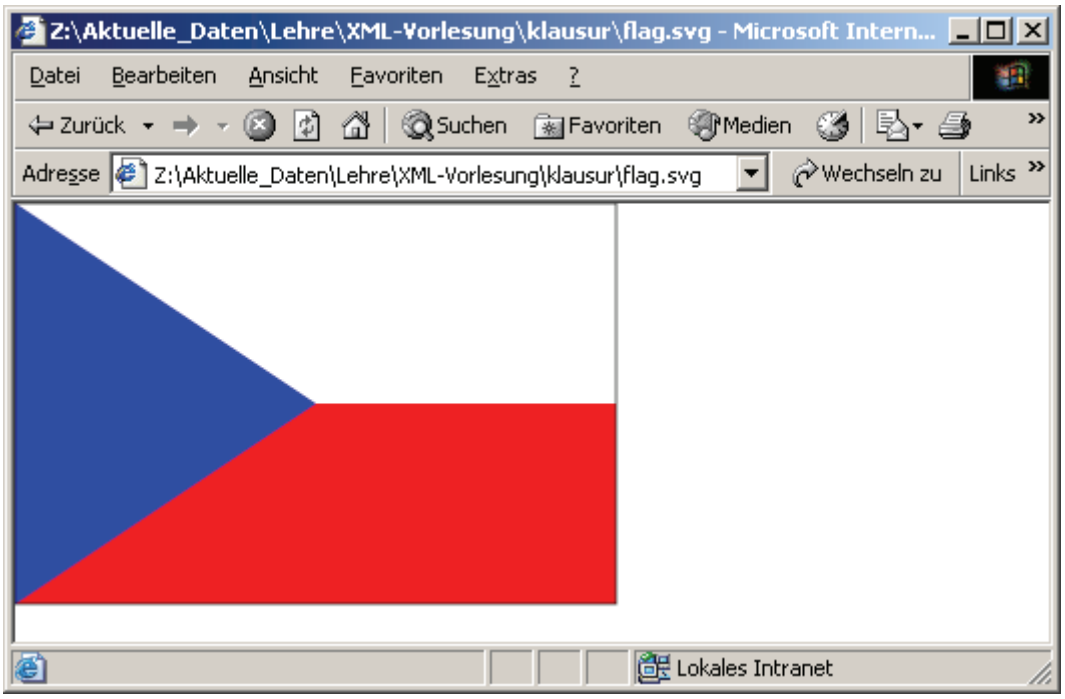
Geben Sie das Ergebnis der untenstehenden SQL/XML-Abfrage an!

```
SELECT XMLELEMENT (
  NAME "SPIEL",
  XMLATTRIBUTES (SCHEINNR, SPIELNR),
  Z1, ' ', Z2, ' ',Z3, ' ', Z4, ' ', Z5, ' ', Z6)
FROM SPIELE
WHERE SCHEINNR = "4711"
```

Aufgabe 9:

Die untenstehende tschechische Nationalflagge hat die Farben *weiß, blau* (im Dreieck) und *rot.* Die Spitze des Dreiecks reicht bis zur halben Breite der Flagge.

Vervollständigen Sie die Lücken im SVG-Dokument unten!




```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg">
```

```
  <rect x="0" y="0" height="200" width="300"
      fill="none" stroke="black" />
</svg>
```

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

Nachname: Vorname:

Matr.Nr.: Studiengang:

Musterlösung

Bearbeiten Sie alle Aufgaben! Bei Ankreuzaufgaben können mehrere Antworten richtig sein. Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 120 Minuten.

Aufgabe	Punkte max.	Punkte erreicht
1	4	
2	2	
3	10	
4	10	
5	8	
6	3	
7	4	
8	4	
9	6	
Summe	**51**	

Aufgabe 1:

Diese Klausur ist ein reines Glücksspiel. Das folgende XML-Dokument beschreibt vereinfacht einen Teil des Spielstands in einem Skatspiel. Es genügt zu wissen: Skat ist ein Kartenspiel für drei Spieler (Hände), das Blatt hat 32 Karten mit *Ass*, *Zehn*, *König*, *Dame*, *Bube*, *Neun*, *Acht*, *Sieben* in den Farben *Kreuz* (♣), *Pik* (♠), *Herz* (♥) und *Karo* (♦). Es werden 30 Karten ausgeteilt, 2 Karten bleiben verdeckt liegen und bilden den Skat. Die *Augen* geben den Wert einer Karte an.

Markieren Sie die Fehler im folgenden, nicht wohlgeformten Dokument!

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="Skat.xsl" ?>
<!DOCTYPE Spiel SYSTEM "skat.dtd">
  <Spiel>
    <Hand Position='vor'>
      <Karte Farbe="Pik" Augen=10>Zehn</Karte>
      <Karte Farbe="Pik" Augen=2>Bube</Karte>
      <Karte Farbe="Karo" Augen=11>Ass</Karte>
    </HAND>
    <Hand Position='mittel'>
      <Karte Farbe="Kreuz" Augen="11">Ass</Karte>
      <Karte Farbe="Herz" Augen="0">Sieben</Karte>
      <Karte Farbe="Karo" Augen="3">Dame</Karte>
    </Hand>
    <Hand Position='hinter'>
      <Karte Farbe="Kreuz" Augen="10">Zehn</Karte>
      <Karte Farbe="Pik" Augen="11">Ass</Karte>
      <Karte Farbe="Karo" Augen="4">Koenig</Karte>
    </Hand>
    <Skat>
      <Karte Farbe = "Herz" Augen = "10">Zehn</Karte>
      <Karte Farbe = "Pik" Augen = "0">Neun</Karte>
    </Skat>
  </Spiel>
```

Aufgabe 2:

Im XML-Dokument aus Aufgabe 1 ist das **Wurzelelement**

( ) **`<?xml ...>`**.

(X) **`<Spiel>`**.

( ) ein nicht sichtbarer Knoten oberhalb von **`<Spiel>`** und **`<?xml ...>`**.

( ) nicht definiert, weil keine DTD angegeben wurde.

Aufgabe 3:

Eine DTD für die Skatspiele aus Aufgabe 1 sieht wie folgt aus.

```
<!-- DTD fuer Skat -->
<!ELEMENT Spiel (Hand, Hand, Hand, Skat)>
<!ATTLIST Spiel Trumpf CDATA #IMPLIED>
<!ELEMENT Hand (Karte)*>
<!ATTLIST Hand Position ( vor | mittel | hinter ) #REQUIRED>
<!ELEMENT Skat (Karte, Karte)>
<!ELEMENT Karte (#PCDATA)>
<!ATTLIST Karte Farbe ( Kreuz | Pik | Herz | Karo ) #REQUIRED>
<!ATTLIST Karte Augen CDATA #REQUIRED>
```

(a) Wie viele Karten hat eine **Hand** mindestens?

Null

(b) Welche **Attribute** sind optional?

Trumpf

(c) Gibt es eine elegantere Methode in einer DTD, um zu erreichen, daß es im Element **<Spiel>** genau drei Hände gibt? Wenn ja, wie lautet die Angabe?

Nein

(d) Gibt es ein Element, für das **gemischter Inhalt** erlaubt ist? Wenn ja, für welches?

Nein

(e) Das **Augen**-Attribut aus **Karte** darf nicht weggelassen werden und sollte eigentlich nur die Integer-Werte 11, 10, 4, 3, 2 oder 0 annehmen. Ändern Sie die Attributdefinition, damit nur diese Werte als gültig (valid) beim Parsen akzeptiert werden?

<!ATTLIST Karte Augen (11 | 10 | 4 | 3 | 2 | 0) #REQUIRED>

Aufgabe 4:

Wie gesagt: reines Glücksspiel. Geben Sie ein gültiges XML-Dokument (einen Lottoschein) zu folgender XML Schema Definition an. Der Lottoschein soll das **Art**-Attribut **"Normal-schein"** haben.

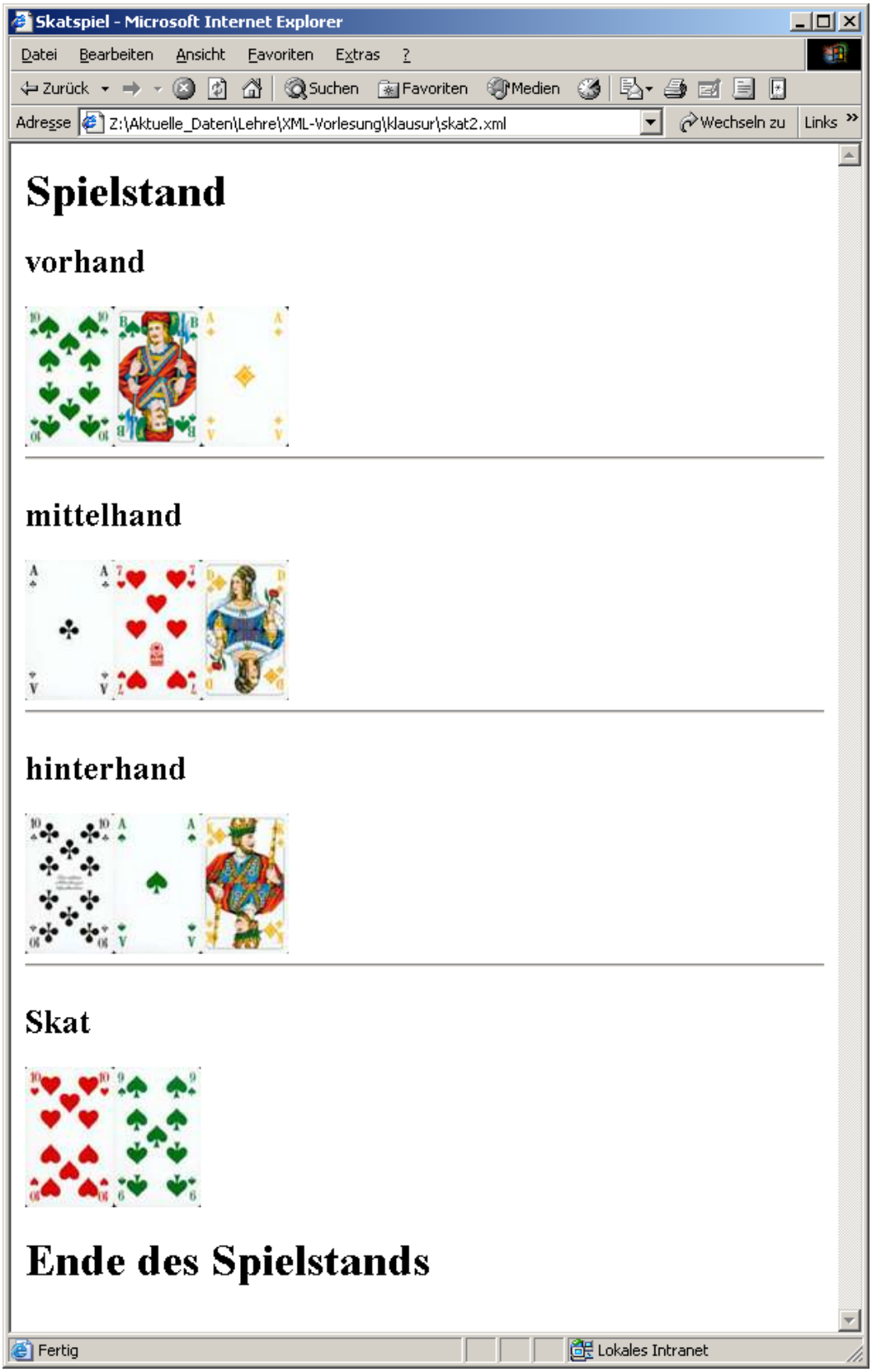
```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:annotation>
  <xsd:documentation xml:lang="de-DE">Schema fuer einen
  vereinfachten Lottoschein.</xsd:documentation>
</xsd:annotation>
<xsd:element name="Lottoschein" type="LScheint" />
<xsd:complexType name="LScheint">
  <xsd:sequence>
    <xsd:element name="Felder" type="Feldert" />
    <xsd:element name="Los-Nummer" type="xsd:integer" />
    <xsd:element name="ersterZiehungstag" type="xsd:string" />
  </xsd:sequence>
  <xsd:attribute name="Art" type="xsd:string"
                 use="required" />
  <xsd:attribute name="LaufzeitWochen" type="LZt"
                 use="optional" default="1" />
  <xsd:attribute name="Ziehungstage" type="xsd:string"
                 use="optional" default="Sa" />
</xsd:complexType>
<xsd:complexType name="Feldert">
  <xsd:sequence>
    <xsd:element name="Lottofeld" type="LFt"
      minOccurs="1" maxOccurs="12" />
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="LFt">
  <xsd:sequence>
    <xsd:element name="Z" type="lottoZahl"
      minOccurs="6" maxOccurs="6" />
  </xsd:sequence>
  <xsd:attribute name="Nr" type="xsd:integer" use="required" />
</xsd:complexType>
<xsd:simpleType name="lottoZahl">
  <xsd:restriction base="xsd:integer">
```

```
    <xsd:minExclusive value="0" />
    <xsd:maxInclusive value="49" />
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="LZt">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="1"/>
    <xsd:enumeration value="2"/>
    <xsd:enumeration value="3"/>
    <xsd:enumeration value="4"/>
    <xsd:enumeration value="Dauer"/>
  </xsd:restriction>
</xsd:simpleType>
</xsd:schema>
```

```
<?xml version="1.0"?>
<Lottoschein Art="Normalschein" >
      <Felder>
         <Lottofeld Nr="1">
           <Z>2</Z>
           <Z>7</Z>
           <Z>12</Z>
           <Z>23</Z>
           <Z>47</Z>
           <Z>49</Z>
         </Lottofeld>
      </Felder>
      <Los-Nummer>2969571</Los-Nummer>
      <ersterZiehungstag>2006-02-25</ersterZiehungstag>
</Lottoschein>
```

Aufgabe 5:

Die nebenstehende Abbildung zeigt die HTML-Ausgabe des XML-Skatblatts aus Aufgabe 1. Die Ausgabe wurde mit dem Stylesheet unten erzeugt. Vervollständigen Sie das Stylesheet!



```
<?xml version='1.0' ?>
<xsl:stylesheet version='1.0'
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
    <head>
        <title>Skatspiel</title>
    </head>
    <body>
        <h1>_____Spielstand_____________________</h1>
        <xsl:apply-templates />
        <h1>Ende des Spielstands</h1>
    </body>
  </html>
</xsl:template>
```

auch nur **Hand** oder **/Spiel/Hand**

```
<xsl:template match="__Spiel/Hand_____>

  <P><h2><xsl:value-of select="__@Position__"/>hand </h2>
  <xsl:apply-templates />
  <HR/></P>
</xsl:template>

<xsl:template match="Karte">
  <img src="{concat(@Farbe,text(),'.jpg')}"/>
</xsl:template>

<xsl:template match="Skat">
    <P><h2>Skat</h2></P>
    _____<xsl:apply-templates/>_____________
</xsl:template>
</xsl:stylesheet>
```

Aufgabe 6:

Wie müssen die JPG-Dateien heißen, damit das Stylesheet oben die passenden Bilder anzeigt? Nennen Sie die Dateinamen der drei JPG-Dateien der Vorhand im Bild oben.

PikZehn.jpg PikBube.jpg KaroAss.jpg

Aufgabe 7:

Die folgende XQuery soll auf dem Skatblatt (skat2.xml) aus Aufgabe 1 laufen, wobei das Dokument dann natürlich wohlgeformt sein muß. Welche Ausgabe wird geliefert?

```
<Augensummen>
{
  for $hand in fn:doc("skat2.xml")/Spiel/Hand
  return
    <Hand Position="{$hand/@Position}">
    {
      fn:sum($hand/Karte/@Augen)
    }
    </Hand>
}
</Augensummen>
```

```
<Augensummen>
  <Hand Position="vor">23</Hand>
  <Hand Position="mittel">14</Hand>
  <Hand Position="hinter">25</Hand>
</Augensummen>
```

Aufgabe 8:

Gegeben sei die folgende Datenbanktabelle SPIELE mit drei Spielen auf zwei Lottoscheinen.

SCHEINNR	SPIELNR	Z1	Z2	Z3	Z4	Z5	Z6
4711	1	5	7	12	20	31	42
4711	2	4	11	22	23	27	44
9362	1	9	10	30	41	42	49

Geben Sie das Ergebnis der untenstehenden SQL/XML-Abfrage an!

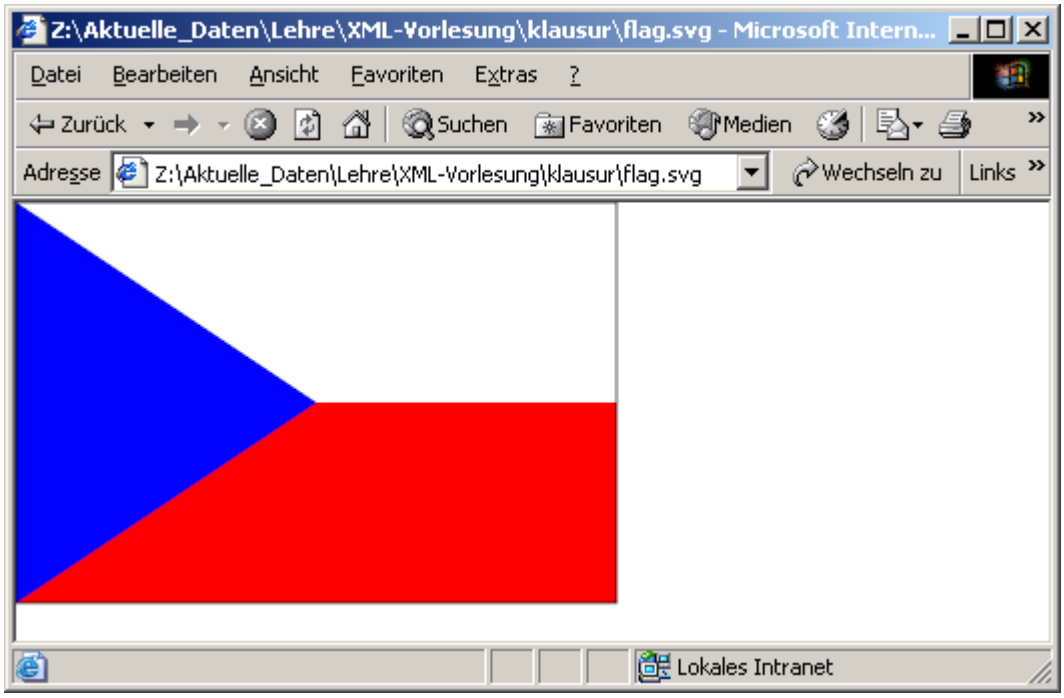
```
SELECT XMLELEMENT (
  NAME "SPIEL",
  XMLATTRIBUTES (SCHEINNR, SPIELNR),
  Z1, ' ', Z2, ' ',Z3, ' ', Z4, ' ', Z5, ' ', Z6)
FROM SPIELE
WHERE SCHEINNR = "4711"
```

```
<SPIEL SCHEINNR="4711" SPIELNR="1">5 7 12 20 31 42</SPIEL>
<SPIEL SCHEINNR="4711" SPIELNR="2">4 11 22 23 27 44</SPIEL>
```

Aufgabe 9:

Die untenstehende tschechische Nationalflagge hat die Farben *weiß*, *blau* (im Dreieck) und *rot*. Die Spitze des Dreiecks reicht bis zur halben Breite der Flagge.

Vervollständigen Sie die Lücken im SVG-Dokument unten!




```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg">
  <rect x="0" y="0" height="100" width="300" fill="white"/>
  <rect x="0" y="100" height="100" width="300" fill="red"/>
  <polyline points="0 0, 150 100, 0 200" fill="blue"/>
```

Kommata spielen keine Rolle

```
  <rect x="0" y="0" height="200" width="300"
      fill="none" stroke="black" />
</svg>
```

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

Nachname: Vorname:

Matr.Nr.: Studiengang:

Bearbeiten Sie alle Aufgaben! Bei Ankreuzaufgaben können mehrere Antworten richtig sein. Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 120 Minuten.

Aufgabe	**Punkte max.**	**Punkte erreicht**
1	4	
2	2 + 2 + 2 + 2 + 2	
3	6 + 4 + 2	
4	6	
5	6	
6	6	
7	6	
Summe	**50**	

Aufgabe 1:

Fußball-Weltmeisterschaft und kein Ende! Betrachten Sie das Dokument *gruppenspiele.xml* aus der „deutschen Gruppe“ in der Vorrunde.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="GStyle.xsl" ?>
<!DOCTYPE Gruppe SYSTEM "gruppenspiele.dtd">
<Gruppe GruppenId="A">
  <Mannschaften>
    <Team>Deutschland</Team>
    <Team>Polen</Team>
    <Team>Ecuador</Team>
    <Team>Costa Rica</Team>
  </Mannschaften>
  <Begegnungen>
    <Spiel Tag="09.06.2006" Zeit="18:00">
      <Heim>Deutschland</Heim><Gast>Costa Rica</Gast>
      <Ergebnis Tore="4:2" Punkte="3:0" />
    </Spiel>
    <Spiel Tag="09.06.2006" Zeit="21:00">
      <Heim>Polen</Heim><Gast>Ecuador</Gast>
      <Ergebnis Tore="0:2" Punkte="0:3" />
    </Spiel>
    <Spiel Tag="14.06.2006" Zeit="21:00">
      <Heim>Deutschland</Heim><Gast>Polen</Gast>
      <Ergebnis Tore="1:0" Punkte="3:0" />
    </Spiel>
    <Spiel Tag="15.06.2006" Zeit="15:00">
      <Heim>Ecuador</Heim><Gast>Costa Rica</Gast>
      <Ergebnis Tore="3:0" Punkte="3:0" />
    </Spiel>
    <Spiel Tag="20.06.2006" Zeit="16:00">
      <Heim>Ecuador</Heim><Gast>Deutschland</Gast>
      <Ergebnis Tore="0:3" Punkte="0:3" />
    </Spiel>
    <Spiel Tag="20.06.2006" Zeit="16:00">
      <Heim>Costa Rica</Heim><Gast>Polen</Gast>
      <Ergebnis Tore="1:2" Punkte="0:3" />
    </Spiel>
  </Begegnungen>
</Gruppe>
```

(a) Ist das Dokument wohlgeformt? Wenn nein, markieren Sie Fehler deutlich.

(b) Würde man im Element **`Ergebnis`** die Reihenfolge der Attribute **`Tore`** und **`Punkte`** vertauschen, dann müßte man das in allen Spielen machen?

Aufgabe 2:

Eine DTD für die Gruppenspiele aus Aufgabe 1 sieht wie folgt aus.

```
<!-- DTD fuer die Gruppenspiele -->
<!ELEMENT Gruppe (Mannschaften, Begegnungen)>
  <!ATTLIST Gruppe GruppenId (A | B | C | D | E | F | G) "A">
<!ELEMENT Mannschaften (Team)+>
<!ELEMENT Team (#PCDATA)>
<!ELEMENT Begegnungen (Spiel)+>
<!ELEMENT Spiel (Heim, Gast, Ergebnis)>
  <!ATTLIST Spiel
    Tag CDATA #REQUIRED
    Zeit CDATA #REQUIRED>
<!ELEMENT Heim (#PCDATA)>
<!ELEMENT Gast (#PCDATA)>
<!ELEMENT Ergebnis EMPTY>
<!ATTLIST Ergebnis
  Tore CDATA #IMPLIED
  Punkte CDATA #IMPLIED>
```

(a) Wie könnte man bestimmen, daß **`Mannschaften`** genau vier **`Team`**-Elemente hat?

(b) Welche **Attribute** sind optional?

(c) Gibt es ein leeres Element? Wenn ja, welches? Dürfen leere Elemente Attribute haben?

(d) Gibt es ein Element, für das **gemischter Inhalt** erlaubt ist? Wenn ja, für welches?

(e) Geben Sie in der DTD bei **`Team`** ein Attribut **`iso`** an, das vom Typ **`ID`** ist. Fügen Sie für die Elemente **`Gast`** und **`Heim`** ein Attribut **`isoref`** hinzu, das vom Typ **`IDREF`** ist. Die Attributwerte hier wären z.B. die Nationencodes *de*, *pl*, *ec*, *cr*.

Aufgabe 3:

Ein XML-Schema zum Gruppenspiele-Dokument sieht wie folgt aus.

(a) Füllen Sie die Lücken, wobei eine Gruppe genau 4 Teams hat und jeder gegen jeden genau einmal spielt!

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:element name="Gruppe" type="GruppeT"/>

  <xsd:complexType name="_____________________________">
    <xsd:sequence>
      <xsd:element name="________________" type="__________________"/>
      <xsd:element name="Begegnungen" type="BegegnungenT"/>
    </xsd:sequence>
    <xsd:attribute name="GruppenId" default="A">
      <xsd:simpleType>
        <xsd:_____________________________ base="xsd:string">
          <xsd:enumeration value="A"/>
          <xsd:enumeration value="B"/>
          <xsd:enumeration value="C"/>
          <xsd:enumeration value="D"/>
          <xsd:enumeration value="E"/>
          <xsd:enumeration value="F"/>
          <xsd:enumeration value="G"/>
          <xsd:enumeration value="H"/>
        </xsd:_______________________>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>

  <xsd:complexType name="MannschaftenT">
    <xsd:sequence>
      <xsd:element name="Team" type="xsd:string"
            minOccurs="________" maxOccurs="__________"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="BegegnungenT">
    <xsd:sequence>
      <xsd:element name="Spiel" type="SpielT"
          minOccurs="__________" maxOccurs="____________"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="SpielT">
    <xsd:sequence>
      <xsd:element name="Heim" type="xsd:string"/>
      <xsd:element name="Gast" type="xsd:string"/>
      <xsd:element name="Ergebnis">
        <xsd:_________________________>
          <xsd:attribute name="Tore" type="xsd:string"/>
          <xsd:attribute name="Punkte" type="xsd:string"/>
        </xsd:_________________________>
```

```
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="Tag" type="xsd:string" use="required"/>
    <xsd:attribute name="Zeit" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:schema>
```

(b) Man erweitere das XML-Schema so, daß man in *gruppenspiele.xml* alle acht Gruppen aufnehmen kann. Nennen Sie das neue Wurzelelement **`Gruppen`**. Geben Sie nur die neuen Teile an.

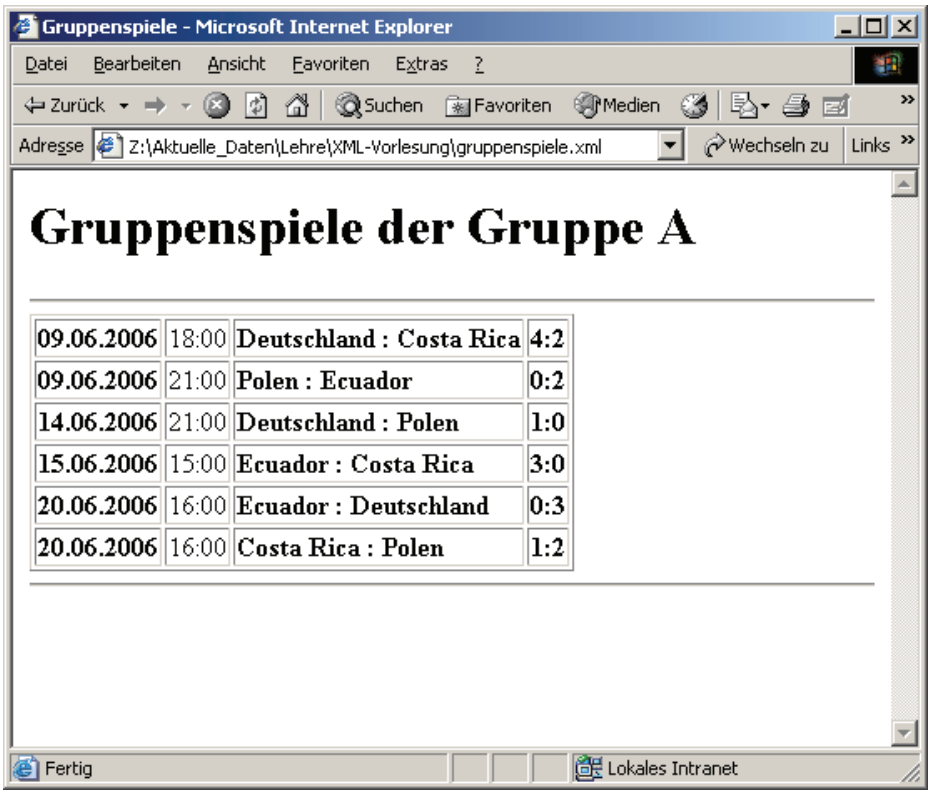
(c) Hätten wir auch im XML-Schema die ID und IDREF Attribute aufgenommen, wären dann unsinnige Spiele wie z.B.

```
<Spiel Tag="09.06.2006" Zeit="18:00">
  <Heim isoref="de">Deutschland</Heim>
  <Gast isoref="de">Deutschland</Gast>
  <Ergebnis .../>
</Spiel>
```

unmöglich? Begründung!

Aufgabe 4:

Die untenstehende Abbildung zeigt die HTML-Ausgabe des XML-Dokuments zu den Gruppenspielen aus Aufgabe 1. Die Ausgabe wurde mit dem Stylesheet unten erzeugt. Vervollständigen Sie das Stylesheet!




```
<?xml version='1.0' standalone='yes'?>
<xsl:stylesheet version='1.0'
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
    <head><title>Gruppenspiele</title></head>
    <body>
      <h1>Gruppenspiele der Gruppe
            <xsl:value-of select="//Gruppe/@GruppenId"/></h1>
      <xsl:apply-templates select="//Begegnungen" />
    </body>
  </html>
</xsl:template>
```

```
<xsl:template match="Begegnungen">
  <hr/>
  <table border="1">
    <xsl:for-each select="_____________________________">
      <tr>
        <td><b><xsl:value-of select="_______________"/></b></td>
        <td><xsl:value-of select="___________________"/></td>
        <td><b><xsl:value-of select="_____________"/> :
            <xsl:value-of select="___________________"/></b>
        </td>
        <td>
          <b><xsl:value-of select="_____________________________"/></b>
        </td>
      </tr>
    </xsl:for-each>
  </table>
  <hr/>
</xsl:template>
</xsl:stylesheet>
```

Aufgabe 5:

Die folgende XQuery soll auf dem Dokument **`gruppenspiele.xml`** aus Aufgabe 1 laufen, wobei das Dokument dann natürlich wohlgeformt sein muß. Welche Ausgabe wird geliefert?

Hinweis: **fn:substring**(*Quellstring*, *Start*, *L*) liefert in einer Zeichenkette *Quellstring* die Teilkette, die bei Zeichenposition *Start* beginnt und *L* Zeichen enthält. Das erste Zeichen in *Quellstring* hat Position 1. **fn:number** wandelt einen String in eine Zahl.

```
<Punktestaende>
{
  let $doc := fn:doc("gruppenspiele.xml")
  for $land in $doc/Gruppe/Mannschaften/Team
  where $land="Deutschland"
  return
    <Pstand>
      <Land>{$land}</Land>
      <Punkte>
      {
        fn:sum(
          for $match in $doc/Gruppe/Begegnungen/Spiel
          return
          if ($match/Heim = $land)
          then fn:number(fn:substring($match/Ergebnis/@Punkte,1,1))
          else
            if ($match/Gast = $land)
            then fn:number(fn:substring(
                              $match/Ergebnis/@Punkte,3,1))
            else ()
        )
      }
      </Punkte>
    </Pstand>
}
</Punktestaende>
```

Aufgabe 6:

Gegeben sei die folgende Datenbanktabelle mit Namen **Gruppenstand**.

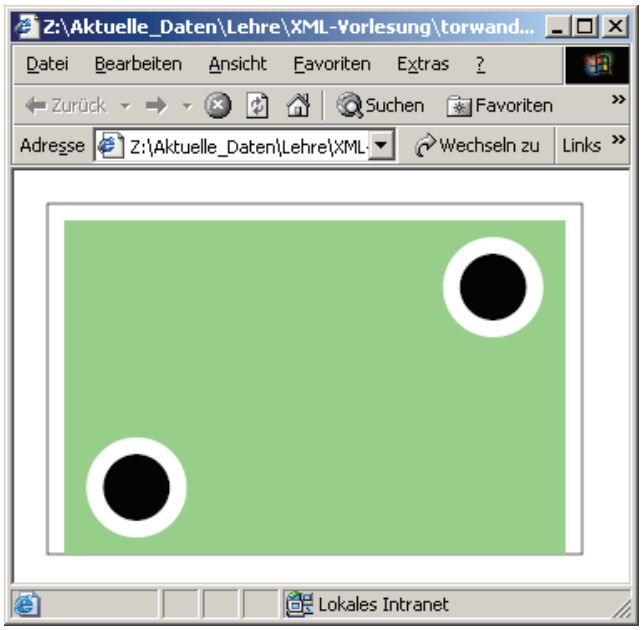
LAND	SPIELE	G	U	V	TORE	PUNKTE
England	3	2	1	0	5:2	7
Schweden	3	1	2	0	3:2	5
Paraguay	3	1	0	2	2:2	3
Trinidad und Tobago	3	0	1	2	0:4	1

Geben Sie das Ergebnis der untenstehenden SQL/XML-Abfrage an!

```
SELECT XMLELEMENT (
  NAME "Punktestand",
  XMLELEMENT (NAME "Land", g.LAND),
  XMLELEMENT (NAME "Punkte",
    XMLATTRIBUTES (g.TORE AS "Torverhaeltnis"),
    g.Punkte))
FROM Gruppenstand g
WHERE g.V = 0
```

Aufgabe 7:

Der Renner in diesem WM-Sommer: animierte Torwände. Tragen Sie die Bewegungsrichtung der Löcher in den Screenshot ein. Füllen Sie die Lücken im SVG-Dokument aus!




```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg">
  <_____________>
    <circle id="_____________" r="25"
        style="fill:black;stroke:white;stroke-width:10"/>
  </_____________>
  <rect x="20" y="20" width="320" height="210"
        fill="white" stroke="black"/>
  <rect x="30" y="30" width="300" height="200" fill="lightgreen" />
  <use y="70" xlink:href="#loch">
    <animate attributeName="_____" values="70;290;70"
        start="0s" dur="6s" repeatCount="indefinite"/>
  </use>
  <use y="190" xlink:href="#loch">
    <animate attributeName="_____" values="_____________________"
        start="0s" dur="6s" repeatCount="indefinite"/>
  </use>
</svg>
```

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

Musterlösung

Nachname: Vorname:

Matr.Nr.: Studiengang:

Bearbeiten Sie alle Aufgaben! Bei Ankreuzaufgaben können mehrere Antworten richtig sein. Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 120 Minuten.

Aufgabe	Punkte max.	Punkte erreicht
1	4	
2	2 + 2 + 2 + 2 + 2	
3	6 + 4 + 2	
4	6	
5	6	
6	6	
7	6	
Summe	**50**	

Aufgabe 1:

Fußball-Weltmeisterschaft und kein Ende! Betrachten Sie das Dokument *gruppenspiele.xml* aus der „deutschen Gruppe“ in der Vorrunde.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="GStyle.xsl" ?>
<!DOCTYPE Gruppe SYSTEM "gruppenspiele.dtd">
<Gruppe GruppenId="A">
  <Mannschaften>
    <Team>Deutschland</Team>
    <Team>Polen</Team>
    <Team>Ecuador</Team>
    <Team>Costa Rica</Team>
  </Mannschaften>
  <Begegnungen>
    <Spiel Tag="09.06.2006" Zeit="18:00">
      <Heim>Deutschland</Heim><Gast>Costa Rica</Gast>
      <Ergebnis Tore="4:2" Punkte="3:0" />
    </Spiel>
    <Spiel Tag="09.06.2006" Zeit="21:00">
      <Heim>Polen</Heim><Gast>Ecuador</Gast>
      <Ergebnis Tore="0:2" Punkte="0:3" />
    </Spiel>
    <Spiel Tag="14.06.2006" Zeit="21:00">
      <Heim>Deutschland</Heim><Gast>Polen</Gast>
      <Ergebnis Tore="1:0" Punkte="3:0" />
    </Spiel>
    <Spiel Tag="15.06.2006" Zeit="15:00">
      <Heim>Ecuador</Heim><Gast>Costa Rica</Gast>
      <Ergebnis Tore="3:0" Punkte="3:0" />
    </Spiel>
    <Spiel Tag="20.06.2006" Zeit="16:00">
      <Heim>Ecuador</Heim><Gast>Deutschland</Gast>
      <Ergebnis Tore="0:3" Punkte="0:3" />
    </Spiel>
    <Spiel Tag="20.06.2006" Zeit="16:00">
      <Heim>Costa Rica</Heim><Gast>Polen</Gast>
      <Ergebnis Tore="1:2" Punkte="0:3" />
    </Spiel>
  </Begegnungen>
</Gruppe>
```

(a) Ist das Dokument wohlgeformt? Wenn nein, markieren Sie Fehler deutlich.

Ja, keine Fehler.

(b) Würde man im Element **`Ergebnis`** die Reihenfolge der Attribute **`Tore`** und **`Punkte`** vertauschen, dann müßte man das in allen Spielen machen?

Nein, muß nicht in allen Elementen vertauscht werden.

Aufgabe 2:

Eine DTD für die Gruppenspiele aus Aufgabe 1 sieht wie folgt aus.

```
<!-- DTD fuer die Gruppenspiele -->
<!ELEMENT Gruppe (Mannschaften, Begegnungen)>
  <!ATTLIST Gruppe GruppenId (A | B | C | D | E | F | G) "A">
<!ELEMENT Mannschaften (Team)+>
<!ELEMENT Team (#PCDATA)>
<!ELEMENT Begegnungen (Spiel)+>
<!ELEMENT Spiel (Heim, Gast, Ergebnis)>
  <!ATTLIST Spiel
    Tag CDATA #REQUIRED
    Zeit CDATA #REQUIRED>
<!ELEMENT Heim (#PCDATA)>
<!ELEMENT Gast (#PCDATA)>
<!ELEMENT Ergebnis EMPTY>
<!ATTLIST Ergebnis
  Tore CDATA #IMPLIED
  Punkte CDATA #IMPLIED>
```

(a) Wie könnte man bestimmen, daß **Mannschaften** genau vier **Team**-Elemente hat?

<!ELEMENT Mannschaften (Team, Team, Team, Team) >

(b) Welche **Attribute** sind optional?

GruppenId, Tore und Punkte

(c) Gibt es ein leeres Element? Wenn ja, welches? Dürfen leere Elemente Attribute haben?

Ja, Ergebnis ist ein leeres Element.

Ja, leere Elemente dürfen Attribute haben.

(d) Gibt es ein Element, für das **gemischter Inhalt** erlaubt ist? Wenn ja, für welches?

Nein

(e) Geben Sie in der DTD bei **Team** ein Attribut **iso** an, das vom Typ **ID** ist. Fügen Sie für die Elemente **Gast** und **Heim** ein Attribut **isoref** hinzu, das vom Typ **IDREF** ist. Die Attributwerte hier wären z.B. die Nationencodes *de*, *pl*, *ec*, *cr*.

<!ATTLIST Team iso ID #REQUIRED> **#IMPLIED auch erlaubt**

<!ATTLIST Heim isoref IDREF #REQUIRED> **oder #IMPLIED**

<!ATTLIST Gast isoref IDREF #REQUIRED> **oder #IMPLIED**

Aufgabe 3:

Ein XML-Schema zum Gruppenspiele-Dokument sieht wie folgt aus.

(a) Füllen Sie die Lücken, wobei eine Gruppe genau 4 Teams hat und jeder gegen jeden genau einmal spielt!

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:element name="Gruppe" type="GruppeT"/>

  <xsd:complexType name="GruppeT">
    <xsd:sequence>
      <xsd:element name="Mannschaften" type="MannschaftenT"/>
      <xsd:element name="Begegnungen" type="BegegnungenT"/>
    </xsd:sequence>
    <xsd:attribute name="GruppenId" default="A">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="A"/>
          <xsd:enumeration value="B"/>
          <xsd:enumeration value="C"/>
          <xsd:enumeration value="D"/>
          <xsd:enumeration value="E"/>
          <xsd:enumeration value="F"/>
          <xsd:enumeration value="G"/>
          <xsd:enumeration value="H"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>

  <xsd:complexType name="MannschaftenT">
    <xsd:sequence>
      <xsd:element name="Team" type="xsd:string"
            minOccurs="4" maxOccurs="4"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="BegegnungenT">
    <xsd:sequence>
      <xsd:element name="Spiel" type="SpielT"
          minOccurs="6" maxOccurs="6"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="SpielT">
    <xsd:sequence>
      <xsd:element name="Heim" type="xsd:string"/>
      <xsd:element name="Gast" type="xsd:string"/>
      <xsd:element name="Ergebnis">
        <xsd:complexType>
          <xsd:attribute name="Tore" type="xsd:string"/>
```

```
          <xsd:attribute name="Punkte" type="xsd:string"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="Tag" type="xsd:string" use="required"/>
    <xsd:attribute name="Zeit" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:schema>
```

(b) Man erweitere das XML-Schema so, daß man in *gruppenspiele.xml* alle acht Gruppen aufnehmen kann. Nennen Sie das neue Wurzelelement **Gruppen**. Geben Sie nur die neuen Teile an.

```
<xsd:element name="Gruppen" type="GruppenT"/>
<xsd:complexType name="GruppenT">
  <xsd:sequence>
    <xsd:element name="Gruppe" type="GruppeT"
        minOccurs="1" maxOccurs="8"/>
  </xsd:sequence>
</xsd:complexType>
```

(c) Hätten wir auch im XML-Schema die ID und IDREF Attribute aufgenommen, wären dann unsinnige Spiele wie z.B.

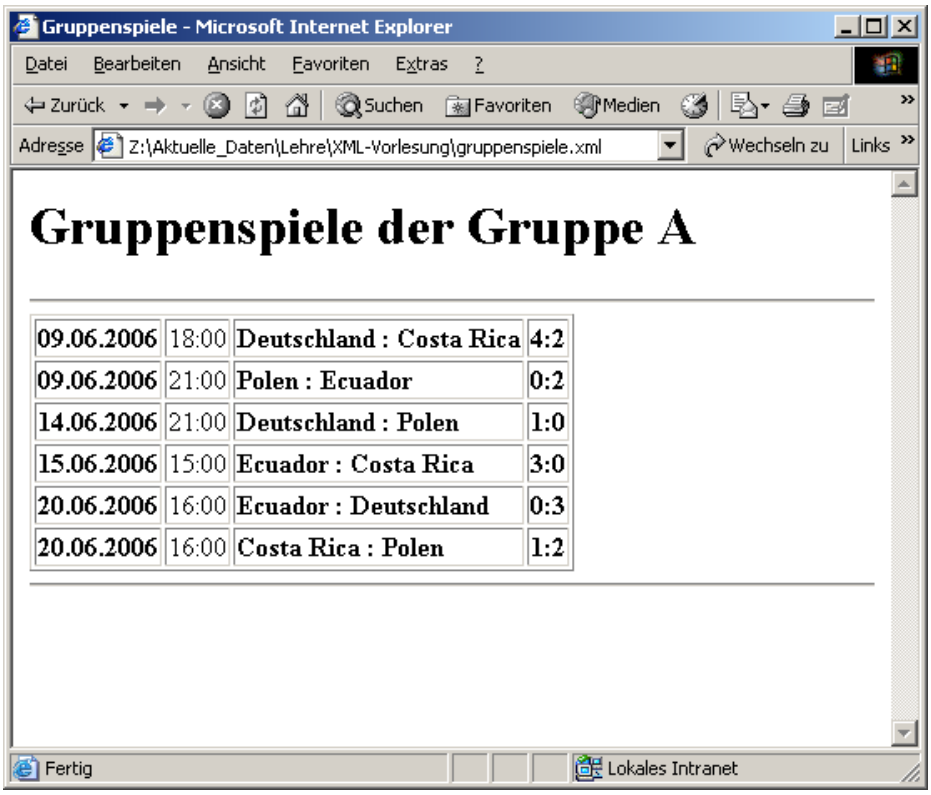
```
<Spiel Tag="09.06.2006" Zeit="18:00">
  <Heim isoref="de">Deutschland</Heim>
  <Gast isoref="de">Deutschland</Gast>
  <Ergebnis .../>
</Spiel>
```

unmöglich? Begründung!

Nein, es wird nur geprüft, ob es eine Mannschaft gibt, die ein Attribut vom Typ ID mit diesem Nationalcode hat.

Aufgabe 4:

Die untenstehende Abbildung zeigt die HTML-Ausgabe des XML-Dokuments zu den Gruppenspielen aus Aufgabe 1. Die Ausgabe wurde mit dem Stylesheet unten erzeugt. Vervollständigen Sie das Stylesheet!




```
<?xml version='1.0' standalone='yes'?>
<xsl:stylesheet version='1.0'
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
    <head><title>Gruppenspiele</title></head>
    <body>
      <h1>Gruppenspiele der Gruppe
            <xsl:value-of select="//Gruppe/@GruppenId"/></h1>
      <xsl:apply-templates select="//Begegnungen" />
    </body>
  </html>
</xsl:template>
```

```
<xsl:template match="Begegnungen">
  <hr/>
  <table border="1">
    <xsl:for-each select="Spiel">
      <tr>
        <td><b><xsl:value-of select="@Tag"/></b></td>
        <td><xsl:value-of select="@Zeit"/></td>
        <td><b><xsl:value-of select="Heim"/> :
            <xsl:value-of select="Gast"/></b>
        </td>
        <td>
          <b><xsl:value-of select="Ergebnis/@Tore"/></b>
        </td>
      </tr>
    </xsl:for-each>
  </table>
  <hr/>
</xsl:template>
</xsl:stylesheet>
```

Aufgabe 5:

Die folgende XQuery soll auf dem Dokument **`gruppenspiele.xml`** aus Aufgabe 1 laufen, wobei das Dokument dann natürlich wohlgeformt sein muß. Welche Ausgabe wird geliefert?

Hinweis: **fn:substring**(*Quellstring*, *Start*, *L*) liefert in einer Zeichenkette *Quellstring* die Teilkette, die bei Zeichenposition *Start* beginnt und *L* Zeichen enthält. Das erste Zeichen in *Quellstring* hat Position 1. **fn:number** wandelt einen String in eine Zahl.

```
<Punktestaende>
{
  let $doc := fn:doc("gruppenspiele.xml")
  for $land in $doc/Gruppe/Mannschaften/Team
  where $land="Deutschland"
  return
    <Pstand>
      <Land>{$land}</Land>
      <Punkte>
      {
        fn:sum(
          for $match in $doc/Gruppe/Begegnungen/Spiel
          return
          if ($match/Heim = $land)
          then fn:number(fn:substring($match/Ergebnis/@Punkte,1,1))
          else
            if ($match/Gast = $land)
            then fn:number(fn:substring(
                              $match/Ergebnis/@Punkte,3,1))
            else ()
        )
      }
      </Punkte>
    </Pstand>
}
</Punktestaende>
```

```
<Punktestaende>
  <Pstand>
    <Land>Deutschland</Land>
    <Punkte>9</Punkte>
  </Pstand>
</Punktestaende>
```

Aufgabe 6:

Gegeben sei die folgende Datenbanktabelle mit Namen **Gruppenstand**.

LAND	SPIELE	G	U	V	TORE	PUNKTE
England	3	2	1	0	5:2	7
Schweden	3	1	2	0	3:2	5
Paraguay	3	1	0	2	2:2	3
Trinidad und Tobago	3	0	1	2	0:4	1

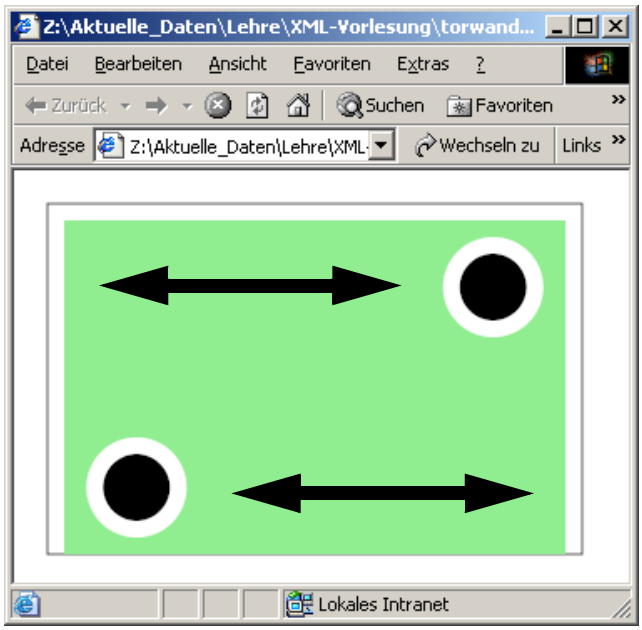
Geben Sie das Ergebnis der untenstehenden SQL/XML-Abfrage an!

```
SELECT XMLELEMENT (
  NAME "Punktestand",
  XMLELEMENT (NAME "Land", g.LAND),
  XMLELEMENT (NAME "Punkte",
    XMLATTRIBUTES (g.TORE AS "Torverhaeltnis"),
    g.Punkte))
FROM Gruppenstand g
WHERE g.V = 0
```

```
<Punktestand>
  <Land>England</Land>
  <Punkte Torverhaeltnis="5:2">7</Punkte>
</Punktestand>
<Punktestand>
  <Land>Schweden</Land>
  <Punkte Torverhaeltnis="3:2">5</Punkte>
</Punktestand>
```

Aufgabe 7:

Der Renner in diesem WM-Sommer: animierte Torwände. Tragen Sie die Bewegungsrichtung der Löcher in den Screenshot ein. Füllen Sie die Lücken im SVG-Dokument aus!




```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg">
  <defs>
    <circle id="loch" r="25"
        style="fill:black;stroke:white;stroke-width:10"/>
  </defs>
  <rect x="20" y="20" width="320" height="210"
        fill="white" stroke="black"/>
  <rect x="30" y="30" width="300" height="200" fill="lightgreen" />
  <use y="70" xlink:href="#loch">
    <animate attributeName="x" values="70;290;70"
        start="0s" dur="6s" repeatCount="indefinite"/>
  </use>
  <use y="190" xlink:href="#loch">
    <animate attributeName="x" values="290;70;290"
        start="0s" dur="6s" repeatCount="indefinite"/>
  </use>
</svg>
```

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

Nachname: Vorname:

Matr.Nr.: Studiengang:

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 90 Minuten.

Aufgabe	Punkte max.	Punkte erreicht
1	2+2	
2	2+2+2+2	
3	6	
4	5	
5	5	
6	6	
7	5	
Summe	**39**	

Als Thema dieser Klausur haben wir vereinfachte Meinungsumfragen zu Politikern und Parteien gewählt. Alle Angaben sind fiktiv und ohne politische Wertung.

Aufgabe 1:

Das folgende Dokument stellt das Ergebnis einer Politikerumfrage dar.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="ergebnisse.xsl" ?>
<Votum ErhobenAm="2007-02-15"
    xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
    xsi:noNamespaceSchemaLocation='ergebnisse.xsd'>

  <Kandidaten>
    <Kandidat Name="Merkel" Partei="CDU">
      <Sympathie>2.2</Sympathie><Kompetenz>3.7</Kompetenz>
      <Vertrauen>4.0</Vertrauen>
    </Kandidat>
    <Kandidat Name="Steinmeier" Partei="SPD">
      <Sympathie>2.1</Sympathie><Kompetenz>3.2</Kompetenz>
      <Vertrauen>1.1</Vertrauen>
    </Kandidat>
    <Kandidat Name="Stoiber" Partei="CSU">
      <Sympathie>-1.2</Sympathie><Kompetenz>3.1</Kompetenz>
      <Vertrauen>0.6</Vertrauen>
    </Kandidat>
    <Kandidat Name="Beck" Partei="SPD">
      <Sympathie>3.8</Sympathie><Kompetenz>2.4</Kompetenz>
      <Vertrauen>2.8</Vertrauen>
    </Kandidat>
    <Kandidat Name="Westerwelle" Partei="FDP">
      <Sympathie>-0.3</Sympathie><Kompetenz>1.8</Kompetenz>
      <Vertrauen>1.5</Vertrauen>
    </Kandidat>
    <Kandidat Name="Kuenast" Partei="Buendnis90DIEGRUENEN">
      <Sympathie>2.3</Sympathie><Kompetenz>2.7</Kompetenz>
      <Vertrauen>2.0</Vertrauen>
    </Kandidat>
  </Kandidaten>
</Votum>
```

(a) Ist das Dokument wohlgeformt? Wenn nein, markieren Sie Fehler deutlich.

(b) Angenommen, die DTD für das Dokument würde es erlauben, könnten dann in einem Dokument wie dem obigen die Elemente **`Sympathie`**, **`Kompetenz`** und **`Vertrauen`** in **`Kandidat`**-Elementen in unterschiedlicher Reihenfolge auftreten?

Aufgabe 2:

Eine DTD für das Umfrage-Dokument aus Aufgabe 1 sieht wie folgt aus.

```
<!-- DTD fuer erhobene Umfragevoten -->
<!ELEMENT Votum (Kandidaten)>
  <!ATTLIST Votum ErhobenAm CDATA #REQUIRED>
<!ELEMENT Kandidaten (Kandidat)+>
<!ELEMENT Kandidat (Sympathie, Kompetenz, Vertrauen)>
  <!ATTLIST Kandidat
    Name CDATA #REQUIRED
    Partei (CDU | CSU | SPD | FDP | Buendnis90DIEGRUENEN |
        DIELINKSPARTEI) #REQUIRED>
<!ELEMENT Sympathie (#PCDATA)>
<!ELEMENT Kompetenz (#PCDATA)>
<!ELEMENT Vertrauen (#PCDATA)>
```

(a) Wie ließe sich ausschließen, daß ein Kandidat aus Versehen zweimal in dem Dokument auftaucht?

(b) Welche **Attribute** sind Pflichtattribute?

(c) Für das Attribut **`Partei`** wurden Alternativen zur Auswahl genannt. Kann man in einer DTD solche Auswahllisten auch für Inhalte von Elementen angeben? Wenn ja, geben Sie ein Beispiel an.

(d) Für das Attribut **`ErhobenAm`** würden wir gerne vorschreiben, dass der Wert ein **Tagesdatum** in einem gültigen Format, z.B. nach ISO 8601 ist. Wie geht das in einer DTD?

Aufgabe 3:

Ein XML-Schema zum Umfrage-Dokument sieht wie folgt aus.

Füllen Sie die Lücken, wobei bei eine Umfrage zwischen 5 und 30 Kandidaten enthalten kann.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="Votum" type="_____________________________"/>

<xsd:complexType name="VotumT">
  <xsd:sequence>
    <xsd:element name="_________________" type="___________________"/>
  </xsd:sequence>
  <xsd:attribute name="ErhobenAm" type="xsd:date" use="required"/>
</xsd:complexType>

<xsd:complexType name="KandidatenT">
  <xsd:sequence>
    <xsd:element name="Kandidat" type="KandidatT"
      minOccurs="_____________" maxOccurs="___________________"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="SkalaT">
  <xsd:restriction base="xsd:_____________________________">
    <xsd:totalDigits value="2"/>
    <xsd:fractionDigits value="1"/>
    <xsd:minInclusive value="-5.0"/>
    <xsd:maxInclusive value="5.0"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="KandidatT">
  <xsd:sequence>
    <xsd:element name="Sympathie" type="SkalaT"/>
    <xsd:element name="Kompetenz" type="SkalaT"/>
    <xsd:element name="Vertrauen" type="SkalaT"/>
  </xsd:sequence>
  <xsd:attribute name="Name" type="xsd:string" use="required"/>
  <xsd:attribute name="Partei" use="required">
    <xsd:simpleType>
      <xsd:_____________________________ base="xsd:string">
        <xsd:enumeration value="CDU"/>
        <xsd:enumeration value="CSU"/>
        <xsd:enumeration value="SPD"/>
        <xsd:enumeration value="FDP"/>
        <xsd:enumeration value="Buendnis90DIEGRUENEN"/>
        <xsd:enumeration value="DIELINKSPARTEI"/>
      </xsd:_____________________________>
    </xsd:simpleType>
  </xsd:attribute>
</xsd:complexType>

</xsd:schema>
```

Aufgabe 4:

Die untenstehende Abbildung zeigt die HTML-Ausgabe des Umfrage-Dokuments aus Aufgabe 1. Die Ausgabe wurde mit dem Stylesheet unten erzeugt. Zu jedem Kandidaten mit Namen *x* lag ein Bild *x*`.jpg` vor. Vervollständigen Sie das Stylesheet!


```
<?xml version='1.0' standalone='yes'?>
<xsl:stylesheet version='1.0'
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
    <head><title>Umfrageergebnisse</title></head>
    <body>
      <h1>Umfrage vom <xsl:value-of
          select="_____"/></h1>
      <xsl:apply-templates select="//Kandidaten" />
```

```
    </body>
  </html>
</xsl:template>

<xsl:template match="_____________________________________">
  <table border="1">
    <tr><th>Kandidat</th><th>Sympathie</th>
      <th>Kompetenz</th><th>Vertrauen</th></tr>
    <xsl:_____________________________________/>
  </table>
</xsl:template>

<xsl:template match="Kandidat">
   <tr>
    <td><img src="{concat(_________________________,text(),'.jpg')}"
          height="60"/></td>
    <td align="center"><xsl:_______________ select="Sympathie"/></td>
    <td align="center"><xsl:_______________ select="Kompetenz"/></td>
    <td align="center"><xsl:_______________ select="Vertrauen"/></td>
  </tr>
</xsl:template>

</xsl:stylesheet>
```

Aufgabe 5:

Die folgende XQuery soll auf dem Umfrage-Dokument **`ergebnisse.xml`** aus Aufgabe 1 laufen, wobei das Dokument dann natürlich wohlgeformt sein muß. Welche Ausgabe wird geliefert?

```
<PListe>
{ for $k in fn:doc("ergebnisse.xml")//Kandidat
  let $w := fn:sum($k/*)
  where $w > 4
  return <Politiker W="{$w}">{fn:string($k/@Name)}</Politiker>
}
</PListe>
```

Aufgabe 6:

Gegeben sei die folgende Datenbanktabelle mit Namen **UMFRAGE**.

NAME	PARTEI	SYMPATHIE	KOMPETENZ	VERTRAUEN
Merkel	CDU	2,2	3,7	4,0
Steinmeier	SPD	2,1	3,2	1,1
Stoiber	CSU	-1,2	3,1	0,6
Beck	SPD	3,8	2,4	2,8
Westerwelle	FDP	-0,3	1,8	1,5
Kuenast	GRUENE	2,3	2,7	2,0

Vervollständigen Sie die folgende SQL/XML-Abrage so, daß die Ausgabe ganz unten geliefert wird. Gesucht sind Politiker, die in allen drei Kategorien nur positive Bewertungen haben. Beachten Sie die Sortierung nach PARTEI und innerhalb von PARTEI nach NAME.

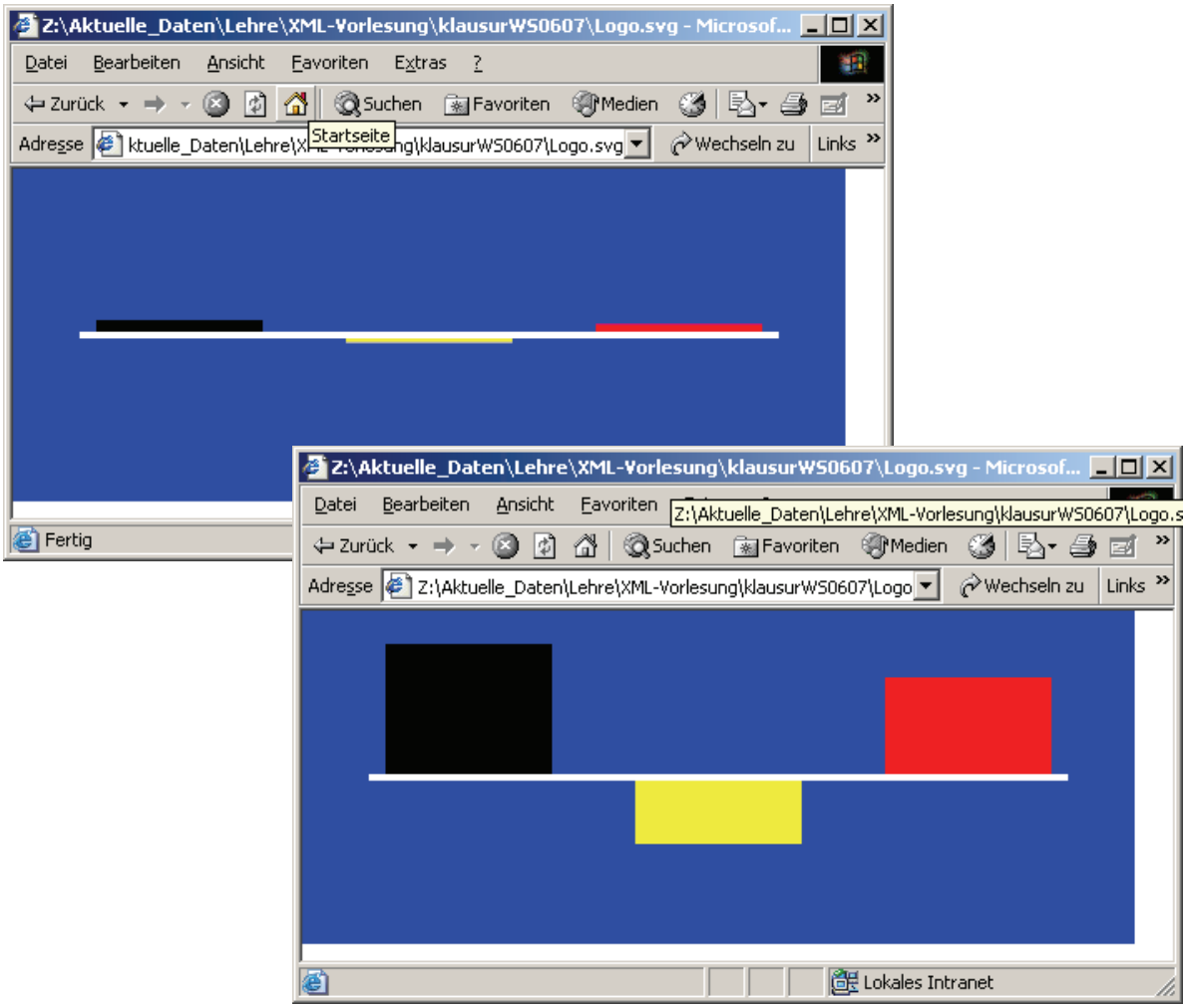
```
SELECT  XMLELEMENT (NAME "_____________________________",
        XMLATTRIBUTES (_________________ AS "_____________________________"),
        ____________________________________)
FROM UMFRAGE u
WHERE ____________________________________________________________________________
ORDER BY _____________________________
```

Ausgabe

```
<Name Partei="CDU">Merkel</Name>
<Name Partei="GRUENE">Kuenast</Name>
<Name Partei="SPD">Beck</Name>
<Name Partei="SPD">Steinmeier</Name>
```

Aufgabe 7:

Es geht um eine animierte Graphik mit drei Balken. Die Screenshots zeigen die Situation kurz nach dem Start und in der Endposition. Füllen Sie die Lücken im SVG-Dokument aus!




```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg">
  <rect x="0" y="0" width="500" height="200" fill="blue" />
  <rect width="100" x="50" fill="black">
    <animate attributeName="__________________" from="0" to="80"
       dur="6s" fill="freeze"/>
    <animate attributeName="__________________" from="100" to="20"
       dur="6s" fill="freeze"/>
  </rect>
    <rect width="100" x="200" y="100" fill="yellow">
    <animate attributeName="__________________" from="0" to="40"
       dur="6s" fill="freeze"/>
  </rect>
  <rect width="100" x="350" fill="red">
    <animate attributeName="__________________" from="0" to="60"
       dur="6s" fill="freeze"/>
    <animate attributeName="__________________" from="100" to="40"
       dur="6s" fill="freeze"/>
  </rect>
  <line x1="40" y1="100" x2="460" y2="100" style="stroke:white;stroke-width:4"/>
</svg>
```

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

MUSTERLÖSUNG

Nachname: Vorname:

Matr.Nr.: Studiengang:

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 90 Minuten.

Aufgabe	Punkte max.	Punkte erreicht
1	2+2	
2	2+2+2+2	
3	6	
4	5	
5	5	
6	6	
7	5	
Summe	**39**	

Als Thema dieser Klausur haben wir vereinfachte Meinungsumfragen zu Politikern und Parteien gewählt. Alle Angaben sind fiktiv und ohne politische Wertung.

Aufgabe 1:

Das folgende Dokument stellt das Ergebnis einer Politikerumfrage dar.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="ergebnisse.xsl" ?>
<Votum ErhobenAm="2007-02-15"
    xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
    xsi:noNamespaceSchemaLocation='ergebnisse.xsd'>

  <Kandidaten>
    <Kandidat Name="Merkel" Partei="CDU">
      <Sympathie>2.2</Sympathie><Kompetenz>3.7</Kompetenz>
      <Vertrauen>4.0</Vertrauen>
    </Kandidat>
    <Kandidat Name="Steinmeier" Partei="SPD">
      <Sympathie>2.1</Sympathie><Kompetenz>3.2</Kompetenz>
      <Vertrauen>1.1</Vertrauen>
    </Kandidat>
    <Kandidat Name="Stoiber" Partei="CSU">
      <Sympathie>-1.2</Sympathie><Kompetenz>3.1</Kompetenz>
      <Vertrauen>0.6</Vertrauen>
    </Kandidat>
    <Kandidat Name="Beck" Partei="SPD">
      <Sympathie>3.8</Sympathie><Kompetenz>2.4</Kompetenz>
      <Vertrauen>2.8</Vertrauen>
    </Kandidat>
    <Kandidat Name="Westerwelle" Partei="FDP">
      <Sympathie>-0.3</Sympathie><Kompetenz>1.8</Kompetenz>
      <Vertrauen>1.5</Vertrauen>
    </Kandidat>
    <Kandidat Name="Kuenast" Partei="Buendnis90DIEGRUENEN">
      <Sympathie>2.3</Sympathie><Kompetenz>2.7</Kompetenz>
      <Vertrauen>2.0</Vertrauen>
    </Kandidat>
  </Kandidaten>
</Votum>
```

(a) Ist das Dokument wohlgeformt? Wenn nein, markieren Sie Fehler deutlich.

Ja, keine Fehler.

(b) Angenommen, die DTD für das Dokument würde es erlauben, könnten dann in einem Dokument wie dem obigen die Elemente **`Sympathie`**, **`Kompetenz`** und **`Vertrauen`** in **`Kandidat`**-Elementen in unterschiedlicher Reihenfolge auftreten?

Ja, wäre möglich.

Aufgabe 2:

Eine DTD für das Votum-Dokument aus Aufgabe 1 sieht wie folgt aus.

```
<!-- DTD fuer erhobene Umfragevoten -->
<!ELEMENT Votum (Kandidaten)>
  <!ATTLIST Votum ErhobenAm CDATA #REQUIRED>
<!ELEMENT Kandidaten (Kandidat)+>
<!ELEMENT Kandidat (Sympathie, Kompetenz, Vertrauen)>
  <!ATTLIST Kandidat
    Name CDATA #REQUIRED
    Partei (CDU | CSU | SPD | FDP | Buendnis90DIEGRUENEN |
         DIELINKSPARTEI) #REQUIRED>
<!ELEMENT Sympathie (#PCDATA)>
<!ELEMENT Kompetenz (#PCDATA)>
<!ELEMENT Vertrauen (#PCDATA)>
```

(a) Wie ließe sich ausschließen, daß ein Kandidat aus Versehen zweimal in dem Dokument auftaucht?

Attribut Name mit Typ ID vereinbaren.Hinweis: ein zusötzliches Attribut id mit Typ ID einzuführen, bringt nichts! (-1 Pkt)

(b) Welche **Attribute** sind Pflichtattribute?

ErhobenAm, Name, Partei

(c) Für das Attribut **`Partei`** wurden Alternativen zur Auswahl genannt. Kann man in einer DTD solche Auswahllisten auch für Inhalte von Elementen angeben? Wenn ja, geben Sie ein Beispiel an.

Nein, geht nicht.

(d) Für das Attribut **`ErhobenAm`** würden wir gerne vorschreiben, dass der Wert ein **Tagesdatum** in einem gültigen Format, z.B. nach ISO 8601 ist. Wie geht das in einer DTD?

Geht gar nicht.

Aufgabe 3:

Ein XML-Schema zum Umfrage-Dokument sieht wie folgt aus.

Füllen Sie die Lücken, wobei bei eine Umfrage zwischen 5 und 30 Kandidaten enthalten kann.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="Votum" type="____VotumT___"/>

<xsd:complexType name="VotumT">
  <xsd:sequence>
    <xsd:element name="Kandidaten" type="KandidatenT"/>
  </xsd:sequence>
  <xsd:attribute name="ErhobenAm" type="xsd:date" use="required"/>
</xsd:complexType>

<xsd:complexType name="KandidatenT">
  <xsd:sequence>
    <xsd:element name="Kandidat" type="KandidatT"
      minOccurs="____5_____" maxOccurs="_____30_____"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="SkalaT">
  <xsd:restriction base="xsd:____decimal_______">
    <xsd:totalDigits value="2"/>
    <xsd:fractionDigits value="1"/>
    <xsd:minInclusive value="-5.0"/>
    <xsd:maxInclusive value="5.0"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="KandidatT">
  <xsd:sequence>
    <xsd:element name="Sympathie" type="SkalaT"/>
    <xsd:element name="Kompetenz" type="SkalaT"/>
    <xsd:element name="Vertrauen" type="SkalaT"/>
  </xsd:sequence>
  <xsd:attribute name="Name" type="xsd:string" use="required"/>
  <xsd:attribute name="Partei" use="required">
    <xsd:simpleType>
      <xsd:___restriction__ base="xsd:string">
        <xsd:enumeration value="CDU"/>
        <xsd:enumeration value="CSU"/>
        <xsd:enumeration value="SPD"/>
        <xsd:enumeration value="FDP"/>
        <xsd:enumeration value="Buendnis90DIEGRUENEN"/>
        <xsd:enumeration value="DIELINKSPARTEI"/>
      </xsd:___restriction___>
    </xsd:simpleType>
  </xsd:attribute>
</xsd:complexType>

</xsd:schema>
```

Aufgabe 4:

Die untenstehende Abbildung zeigt die HTML-Ausgabe des Umfrage-Dokuments aus Aufgabe 1. Die Ausgabe wurde mit dem Stylesheet unten erzeugt. Zu jedem Kandidaten mit Namen *x* lag ein Bild *x*`.jpg` vor. Vervollständigen Sie das Stylesheet!


```
<?xml version='1.0' standalone='yes'?>
<xsl:stylesheet version='1.0'
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
    <head><title>Umfrageergebnisse</title></head>
    <body>
      <h1>Umfrage vom <xsl:value-of
          select="_Votum/@ErhobenAm__"/></h1>
      <xsl:apply-templates select="//Kandidaten" />
```

```
    </body>
  </html>
</xsl:template>

<xsl:template match="__Kandidaten__">
  <table border="1">
    <tr><th>Kandidat</th><th>Sympathie</th>
      <th>Kompetenz</th><th>Vertrauen</th></tr>
    <xsl:__apply-templates_____/>
  </table>
</xsl:template>

<xsl:template match="Kandidat">
   <tr>
    <td><img src="{concat(__@Name____,text(),'.jpg')}"
          height="60"/></td>
    <td align="center"><xsl:value-of select="Sympathie"/></td>
    <td align="center"><xsl:value-of select="Kompetenz"/></td>
    <td align="center"><xsl:value-of select="Vertrauen"/></td>
  </tr>
</xsl:template>

</xsl:stylesheet>
```

Aufgabe 5:

Die folgende XQuery soll auf dem Umfrage-Dokument **`ergebnisse.xml`** aus Aufgabe 1 laufen, wobei das Dokument dann natürlich wohlgeformt sein muß. Welche Ausgabe wird geliefert?

```
<PListe>
{ for $k in fn:doc("ergebnisse.xml")//Kandidat
  let $w := fn:sum($k/*)
  where $w > 4
  return <Politiker W="{$w}">{fn:string($k/@Name)}</Politiker>
}
</PListe>
```

```
<PListe>
  <Politiker W="9.9">Merkel</Politiker>
  <Politiker W="6.4">Steinmeier</Politiker>
  <Politiker W="9">Beck</Politiker>
  <Politiker W="7">Kuenast</Politiker>
</PListe>
```

Aufgabe 6:

Gegeben sei die folgende Datenbanktabelle mit Namen **UMFRAGE**.

NAME	PARTEI	SYMPATHIE	KOMPETENZ	VERTRAUEN
Merkel	CDU	2,2	3,7	4,0
Steinmeier	SPD	2,1	3,2	1,1
Stoiber	CSU	-1,2	3,1	0,6
Beck	SPD	3,8	2,4	2,8
Westerwelle	FDP	-0,3	1,8	1,5
Kuenast	GRUENE	2,3	2,7	2,0

Vervollständigen Sie die folgende SQL/XML-Abfrage so, daß die Ausgabe ganz unten geliefert wird. Gesucht sind Politiker, die in allen drei Kategorien nur positive Bewertungen haben. Beachten Sie die Sortierung nach PARTEI und innerhalb von PARTEI nach NAME.

```
SELECT  XMLELEMENT (NAME "Name",
          XMLATTRIBUTES (u.PARTEI AS "Partei"),
        u.NAME)
FROM UMFRAGE u
WHERE u.SYMPATHIE > 0 AND u.KOMPETENZ > 0 AND u.VERTRAUEN > 0
ORDER BY u.PARTEI, u.NAME
```

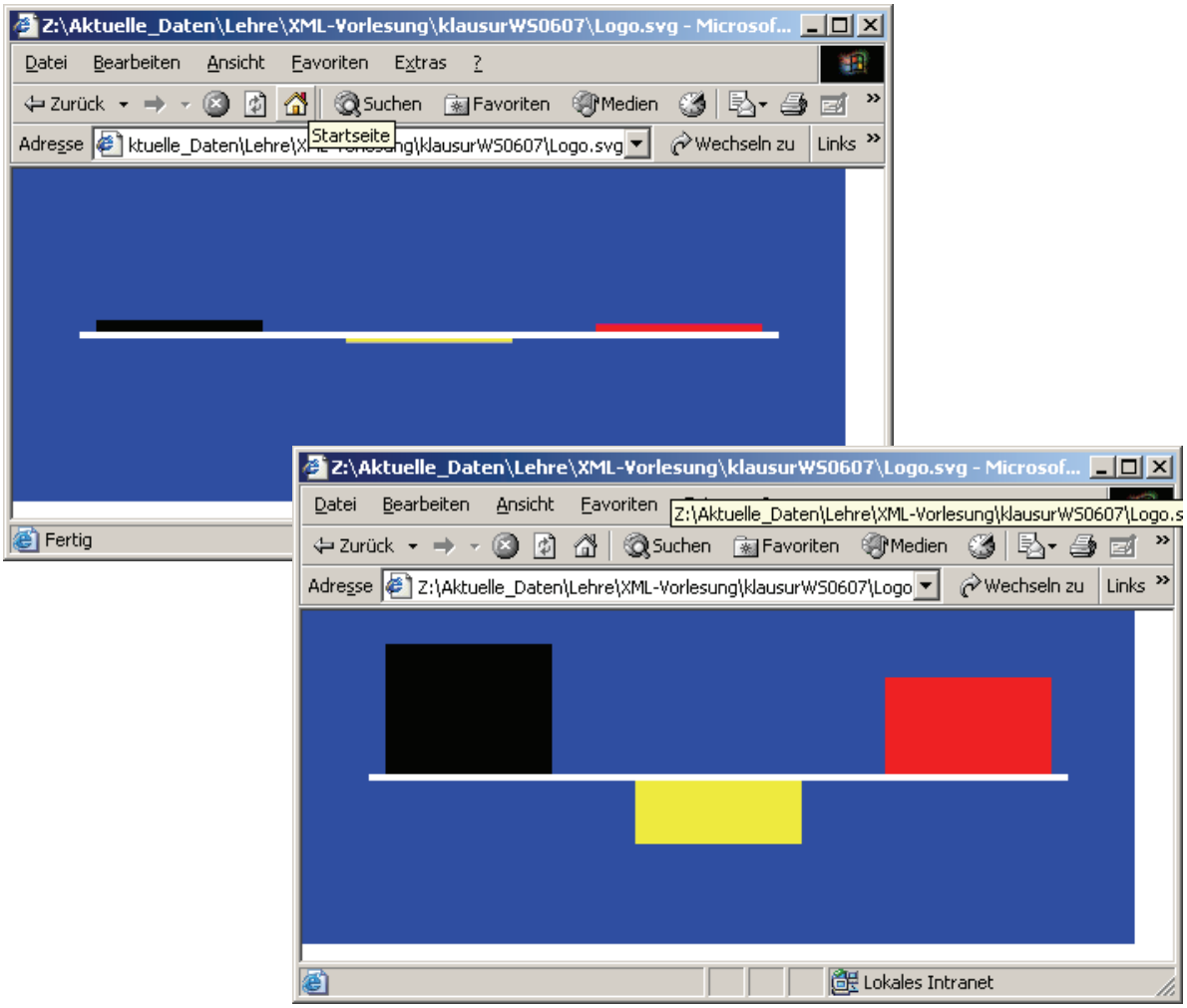
Ausgabe **(*)**

```
<Name Partei="CDU">Merkel</Name>
<Name Partei="GRUENE">Kuenast</Name>
<Name Partei="SPD">Beck</Name>
<Name Partei="SPD">Steinmeier</Name>
```

(*) In der ausgeteilten Klausur waren die ersten beiden Zeilen der Ausgabe vertauscht, weil die Sortierfolge noch auf dem ursprünglichen Parteinamen "Buendnis90DIEGRUENEN" beruhte, was dann von Hand zu "GRUENE" verändert wurde. Dieser Fehler wurde in der Klausur bekanntgegeben.

Aufgabe 7:

Es geht um eine animierte Graphik mit drei Balken. Die Screenshots zeigen die Situation kurz nach dem Start und in der Endposition. Füllen Sie die Lücken im SVG-Dokument aus!




```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg">
  <rect x="0" y="0" width="500" height="200" fill="blue" />
  <rect width="100" x="50" fill="black">
    <animate attributeName="___height___" from="0" to="80"
       dur="6s" fill="freeze"/>
    <animate attributeName="____y____" from="100" to="20"
       dur="6s" fill="freeze"/>
  </rect>
    <rect width="100" x="200" y="100" fill="yellow">
    <animate attributeName="___height___" from="0" to="40"
       dur="6s" fill="freeze"/>
  </rect>
  <rect width="100" x="350" fill="red">
    <animate attributeName="__height__" from="0" to="60"
       dur="6s" fill="freeze"/>
    <animate attributeName="_____y_____" from="100" to="40"
       dur="6s" fill="freeze"/>
  </rect>
  <line x1="40" y1="100" x2="460" y2="100" style="stroke:white;stroke-width:4"/>
</svg>
```

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

Nachname: Vorname:

Matr.Nr.: Studiengang:

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 90 Minuten.

Aufgabe	**Punkte max.**	**Punkte erreicht**
1	3+2	
2	4+2+2+2	
3	8	
4	5+3	
5	6	
6	6	
7	6	
Summe	**49**	

Aufgabe 1:

Das folgende Dokument **urlaub.xml** stellt vereinfacht „Last Minute“ Urlaubsangebote dar.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="urlaub.xsl" ?>
<Angebote
 xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
     xsi:noNamespaceSchemaLocation='urlaub.xsd'
     Abflug="Frankfurt" Tage=7 Personen=2>

  <Angebot Region = "Gran Canaria" Ort="Playa del Ingles">
        <Hotel Sterne="3" Unterkunft="DZ">Sahara Playa</Hotel>
        <Preis Leistung="HP">974</Preis>
  </Angebot>
  <Angebot Region="Gran Canaria" Ort='Playa del Ingles'>
        <Hotel Unterkunft="DZ" Sterne="3">Maritim Playa<Hotel/>
        <Preis Leistung="HP">950</Preis>
  </Angebot>
  <Angebot Region="Ibiza" Ort="Figueretas">
        <Apartment Sterne="3">Apartamentos Lido</Apartment>
        <Preis Leistung="Ue">676</Preis>
  </Angebot>
  <Angebot Region="Antalya" Ort="Side">
        <Hotel Sterne="4" Unterkunft="DZ">Side Star Park</Hotel>
        <Preis Leistung="AI">974</Preis>
  </Angebot>
  <Angebot Region="Kreta" Ort="Malia">
        <Hotel Sterne="4" Unterkunft="DZ">Phaedra Beach</Hotel>
        <Preis Leistung="HP">928</Preis>
  </ANGEBOT>
</Angebote>
```

(a) Ist das Dokument wohlgeformt? Wenn nein, markieren Sie Fehler deutlich.

(b) Eignet sich eines der Attribute als Schlüssel (Typ ID)? Kurze Begründung!

Aufgabe 2:

(a) Die DTD für das Urlaubs-Dokument aus Aufgabe 1 enthält eine Reihe von Fehlern. Markieren Sie diese deutlich.

```
<!-- DTD fuer Urlaubsangebote -->
<!ELEMENT Angebote (Angebot)+>
  <!ATTLIST Angebote
       Abflug CDATA #REQUIRED
       Tage CDATA #REQUIRED
       Personen CDATA #IMPLIED #DEFAULT>
<!ELEMENT Angebot ((Hotel | Apartment), Preis)>
  <!ATTLIST Angebot
       Region CDATA #IMPLIED
       Ort CDATA #REQUIRED>
<!ELEMENT Hotel (#PCDATA)>
  <!ATTLIST Hotel
       Sterne CDATA #REQUIRED
       Unterkunft ( EZ | DZ | Studio | DBZ ) DZ >
<!ELEMENT Apartment (#PCDATA)>
  <!Attlist APARTMENT
       Sterne CDATA #REQUIRED >
<!ELEMENT Preis (#DECIMAL)>
  <!ATTLIST Preis
       Leistung ( Ue | UeF | HP | VP | AI | AIPlus ) Ue >
```

(b) Wenn wie im Fall des Elements **`Hotel`** die beiden Attribute **`Sterne`** und **`Unterkunft`** in dieser Reihenfolge angegeben werden, dann müssen die Attribute auch in dieser Reihenfolge im Dokument stehen. Ist diese Aussage richtig oder falsch?

(c) Angenommen wir wollten die Anzahl der Sterne eines Hotels nicht mit Zahlen (**`Sterne="4"`**) sondern z.B. mit **`Sterne="****"`** angeben. Erlaubt die obige DTD dies?

(d) Wäre es möglich gewesen, beim Attribut **`Leistung`** Übernachtung statt mit **`"Ue"`** auch mit **`"Ü"`** abzukürzen? Ist an dieser Stelle ein Namenstoken erforderlich?

Aufgabe 3:

Ein XML-Schema zum Urlaubs-Dokument sieht wie folgt aus. Dabei haben wir für das Attribut **`Unterkunft`** vereinfachend nur den Typ **`xsd:string`** vorgesehen. Füllen Sie die Lücken, wobei zwischen 0 und 99 Angebote möglich sein sollen.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<xsd:element name="Angebote" type="AngeboteT"/>

<xsd:complexType name="_____________________________">
  <xsd:sequence>
    <xsd:element name="Angebot" type="AngebotT"
    minOccurs="__________" maxOccurs="_________________"/>
  </xsd:sequence>
  <xsd:attribute name="Abflug" type="xsd:string" use="required"/>
  <xsd:attribute name="Tage" type="xsd:decimal" use="required"/>
  <xsd:attribute name="Personen" type="xsd:decimal" use="optional"/>
</xsd:complexType>

<xsd:complexType name="AngebotT">
  <xsd:sequence>
    <xsd:_____________________>
      <xsd:element name="Hotel" type="HotelT"/>
      <xsd:element name="Apartment" type="ApartmentT"/>
    </xsd:_____________________>
    <xsd:element name="Preis">
      <xsd:complexType>
        <xsd:_____________________________>
          <xsd:_____________________ base="xsd:decimal" >
            <xsd:attribute name="Leistung" type="LeistungT"
                  use="optional" default="Ue" />
          </xsd:_____________________>
        </xsd:_____________________________>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
  <xsd:attribute name="_____________________" type="xsd:string"/>
  <xsd:attribute name="_____________________" type="xsd:string"/>
</xsd:complexType>
```

```
<xsd:complexType name="HotelT">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="Sterne" type="xsd:decimal"/>
      <xsd:attribute name="Unterkunft" type="xsd:string"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="ApartmentT">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="Sterne" type="xsd:decimal"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:simpleType name="LeistungT">
  <xsd:_____________________________________ base="xsd:string">
    <xsd:enumeration value="Ue"/>
    <xsd:enumeration value="UeF"/>
    <xsd:enumeration value="HP"/>
    <xsd:enumeration value="VP"/>
    <xsd:enumeration value="AI"/>
    <xsd:enumeration value="AIPlus"/>
  </xsd:_____________________________________>
</xsd:simpleType>

</xsd:schema>
```

Aufgabe 4:

Die untenstehende Abbildung zeigt die HTML-Ausgabe des Urlaubs-Dokuments aus Aufgabe 1. Die Ausgabe wurde mit dem Stylesheet unten erzeugt.

(a) Vervollständigen Sie es!

(b) Erläutern Sie dann ganz unten, warum der Hotel- bzw. Apartmentname in der Ausgabe erscheint, obwohl dafür kein **`<xsl:value-of select="...">`** angegeben wurde.


```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet version='1.0'
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
    <head><title>Last Minute Angebote</title></head>
    <body>
      <h1>Last Minute ab <xsl:value-of
            select="_____"/></h1>
      <ul><xsl:apply-templates select="//Angebot" /></ul>
    </body>
  </html>
</xsl:template>

<xsl:template _____>
  <li><xsl:apply-templates select="Hotel | Apartment"/> in
    <xsl:value-of select="@Ort"/>
    (Region: <xsl:value-of select="@Region"/>)
```

```
    <xsl:apply-templates select="_____________________________"/>
    <xsl:if test="________/@Sterne &gt; 3"><b> Komfort! </b></xsl:if>
  </li>
</xsl:template>

<xsl:template match="Preis">
  Preis: <xsl:value-of select="_______________"/> EUR für
      <xsl:value-of select="@Leistung"/>
  <xsl:if test=". &lt; 800"><b> Schnäppchen! </b></xsl:if>
</xsl:template>

</xsl:stylesheet>
```

Hinweis: Beachten Sie, daß Hotels **und** Apartments das Attribut **`Sterne`** haben!

Erläuterung zu (b) - Warum der Hotel- bzw. Apartmentname in der Ausgabe erscheint.

Aufgabe 5:

Die folgende XQuery soll auf dem Urlaubs-Dokument **`urlaub.xml`** aus Aufgabe 1 laufen, wobei das Dokument dann natürlich wohlgeformt sein muß. Welche Ausgabe wird geliefert?

```
<Komfort>
{ for $a in fn:doc("urlaub.xml")//Angebot
  where $a/*/@Sterne > 3
  return
    <top>
      {fn:string($a/(Hotel | Apartment))} in {fn:string($a/@Ort)}
    </top>
}
</Komfort>
```

Aufgabe 6:

Gegeben sei die folgende Datenbanktabelle mit Namen **ANGEBOTE**.

REGION	ORT	NAME	TYP	STERNE	PREIS	LEISTUNG
Gran Canaria	Playa del Ingles	Sahara Playa	H	3	974	HP
Gran Canaria	Playa del Ingles	Maritim Playa	H	3	950	HP
Ibiza	Figueretas	Apartamentos Lido	A	3	676	Ue
Antalya	Side	Side Star Park	H	4	974	AI
Kreta	Malia	Phaedra Beach	H	4	928	HP

Vervollständigen Sie die folgende SQL/XML-Abfage so, daß die Ausgabe ganz unten geliefert wird.

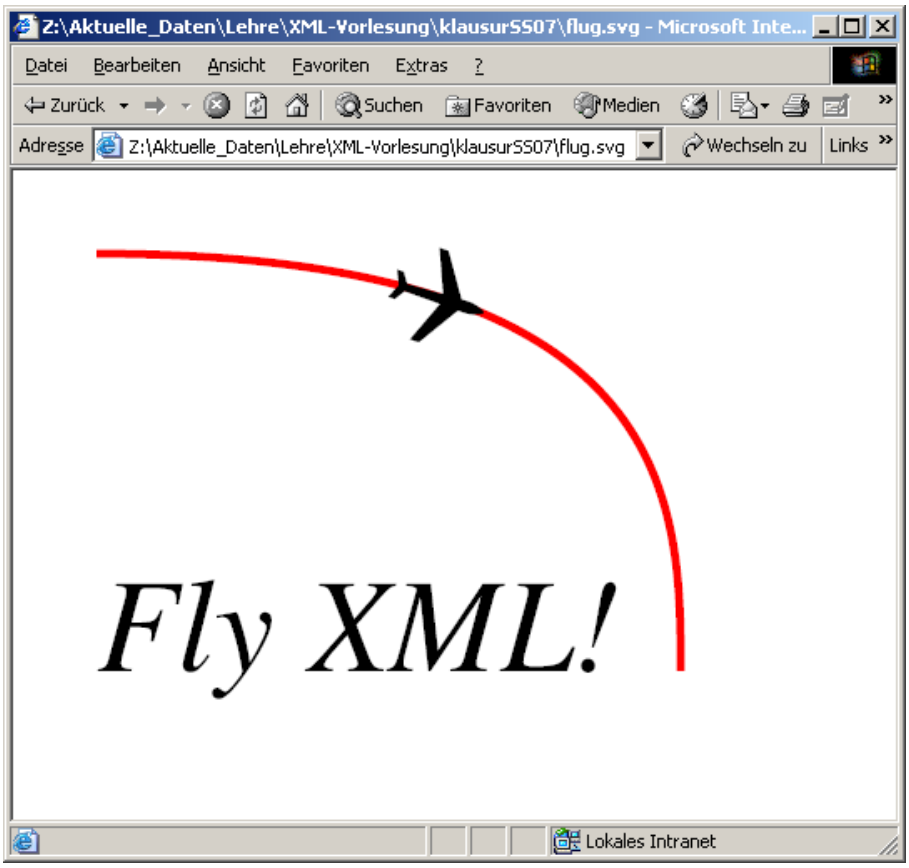
```
SELECT
  XMLELEMENT(NAME "Komfortklasse",
    XMLATTRIBUTES(a.STERNE AS "Sterne"),
    XMLAGG(
      ____________________________________________,
      ____________________________________________________________,
        _____________________________))
FROM ANGEBOTE a
GROUP BY a.STERNE;
```

Ausgabe

```
<Komfortklasse Sterne="3">
  <Angebot Ort="Playa del Ingles" Name="Sahara Playa">974</Angebot>
  <Angebot Ort="Playa del Ingles" Name="Maritim Playa">950</Angebot>
  <Angebot Ort="Figueretas" Name="Apartamentos Lido">676</Angebot>
</Komfortklasse>
<Komfortklasse Sterne="4">
  <Angebot Ort="Side" Name="Side Star Park">974</Angebot>
  <Angebot Ort="Malia" Name="Phaedra Beach">928</Angebot>
</Komfortklasse>
```

Aufgabe 7:

Diese unglaublich elegante, animierte Graphik verwendet die Tatsache, daß es in der Font-Familie Wingdings ein Zeichen gibt, das ein stilisiertes Flugzeug darstellt, das in unserem Fall entlang der Linie fliegt. Für die Lücken im SVG-Dokument geben wir rechts eine Auswahl an. Füllen Sie die Lücken mit den richtigen Nummern!




```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg">
  <path d="M50 50 C 400 50 400 200 400 300" stroke="red"
      stroke-width="_________" fill="________"/>

  <text x="0" y="30" font-family="_______" font-size="80"
      text-anchor="________">&#x51;
    <animateMotion path="M50 50 C 400 50 400 200 400 300"
      dur="_____" rotate="_______" repeatCount="_______" />
  </text>

  <text x="50" y="300" font-family="Times"
    font-size="______" font-style="________">Fly XML!
  </text>
  </svg>
```

① indefinite
② none
③ middle
④ auto
⑤ italic
⑥ Wingdings
⑦ 5
⑧ 80
⑨ 6s

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

Nachname: Vorname:

Matr.Nr.: Studiengang:

MUSTERLÖSUNG

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 90 Minuten.

Aufgabe	Punkte max.	Punkte erreicht
1	3+2	
2	4+2+2+2	
3	8	
4	5+3	
5	6	
6	6	
7	6	
Summe	**49**	

Aufgabe 1:

Das folgende Dokument **urlaub.xml** stellt vereinfacht „Last Minute“ Urlaubsangebote dar.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="urlaub.xsl" ?>
<Angebote
 xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
     xsi:noNamespaceSchemaLocation='urlaub.xsd'
     Abflug="Frankfurt" Tage=7 Personen=2>

  <Angebot Region = "Gran Canaria" Ort="Playa del Ingles">
        <Hotel Sterne="3" Unterkunft="DZ">Sahara Playa</Hotel>
        <Preis Leistung="HP">974</Preis>
  </Angebot>
  <Angebot Region="Gran Canaria" Ort='Playa del Ingles'>
        <Hotel Unterkunft="DZ" Sterne="3">Maritim Playa<Hotel/>
        <Preis Leistung="HP">950</Preis>
  </Angebot>
  <Angebot Region="Ibiza" Ort="Figueretas">
        <Apartment Sterne="3">Apartamentos Lido</Apartment>
        <Preis Leistung="Ue">676</Preis>
  </Angebot>
  <Angebot Region="Antalya" Ort="Side">
        <Hotel Sterne="4" Unterkunft="DZ">Side Star Park</Hotel>
        <Preis Leistung="AI">974</Preis>
  </Angebot>
  <Angebot Region="Kreta" Ort="Malia">
        <Hotel Sterne="4" Unterkunft="DZ">Phaedra Beach</Hotel>
        <Preis Leistung="HP">928</Preis>
  </ANGEBOT>
</Angebote>
```

(a) Ist das Dokument wohlgeformt? Wenn nein, markieren Sie Fehler deutlich.

Nein

(b) Eignet sich eines der Attribute als Schlüssel (Typ ID)? Kurze Begründung!

Nein, da kein Attribut eindeutig ist (Schlüsseleigenschaft hat).

Aufgabe 2:

(a) Die DTD für das Urlaubs-Dokument aus Aufgabe 1 enthält eine Reihe von Fehlern. Markieren Sie diese deutlich.

```
<!-- DTD fuer Urlaubsangebote -->
<!ELEMENT Angebote (Angebot)+>
  <!ATTLIST Angebote
       Abflug CDATA #REQUIRED
       Tage CDATA #REQUIRED
       Personen CDATA #IMPLIED #DEFAULT>
<!ELEMENT Angebot ((Hotel | Apartment), Preis)>
  <!ATTLIST Angebot
       Region CDATA #IMPLIED
       Ort CDATA #REQUIRED>
<!ELEMENT Hotel (#PCDATA)>
  <!ATTLIST Hotel
       Sterne CDATA #REQUIRED
       Unterkunft ( EZ | DZ | Studio | DBZ ) DZ >
<!ELEMENT Apartment (#PCDATA)>
  <!Attlist APARTMENT
       Sterne CDATA #REQUIRED >
<!ELEMENT Preis (#DECIMAL)>
  <!ATTLIST Preis
       Leistung ( Ue | UeF | HP | VP | AI | AIPlus ) Ue >
```

(b) Wenn wie im Fall des Elements **`Hotel`** die beiden Attribute **`Sterne`** und **`Unterkunft`** in dieser Reihenfolge angegeben werden, dann müssen die Attribute auch in dieser Reihenfolge im Dokument stehen. Ist diese Aussage richtig oder falsch?

falsch

(c) Angenommen wir wollten die Anzahl der Sterne eines Hotels nicht mit Zahlen (**`Sterne="4"`**) sondern z.B. mit **`Sterne="*****"`** angeben. Erlaubt die obige DTD dies?

ja

(d) Wäre es möglich gewesen, beim Attribut **`Leistung`** Übernachtung statt mit **`"Ue"`** auch mit **`"Ü"`** abzukürzen? Ist an dieser Stelle ein Namenstoken erforderlich?

Ü statt Ue möglich. Ja, Namenstoken erforderlich.

Aufgabe 3:

Ein XML-Schema zum Urlaubs-Dokument sieht wie folgt aus. Dabei haben wir für das Attribut **Unterkunft** vereinfachend nur den Typ **xsd:string** vorgesehen. Füllen Sie die Lücken, wobei zwischen 0 und 99 Angebote möglich sein sollen.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<xsd:element name="Angebote" type="AngeboteT"/>

<xsd:complexType name="_AngeboteT_">
  <xsd:sequence>
    <xsd:element name="Angebot" type="AngebotT"
    minOccurs="___0___" maxOccurs="____99___"/>
  </xsd:sequence>
  <xsd:attribute name="Abflug" type="xsd:string" use="required"/>
  <xsd:attribute name="Tage" type="xsd:decimal" use="required"/>
  <xsd:attribute name="Personen" type="xsd:decimal" use="optional"/>
</xsd:complexType>

<xsd:complexType name="AngebotT">
  <xsd:sequence>
    <xsd:___choice____>
      <xsd:element name="Hotel" type="HotelT"/>
      <xsd:element name="Apartment" type="ApartmentT"/>
    </xsd:___choice__>
    <xsd:element name="Preis">
      <xsd:complexType>
        <xsd:__simpleContent___>
          <xsd:__extension___ base="xsd:decimal" >
            <xsd:attribute name="Leistung" type="LeistungT"
                  use="optional" default="Ue" />
          </xsd:___extension__>
        </xsd:____simpleContent___>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
  <xsd:attribute name="__Region____" type="xsd:string"/>
  <xsd:attribute name="__Ort__" type="xsd:string"/>
```

```
</xsd:complexType>

<xsd:complexType name="HotelT">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="Sterne" type="xsd:decimal"/>
      <xsd:attribute name="Unterkunft" type="xsd:string"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="ApartmentT">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="Sterne" type="xsd:decimal"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:simpleType name="LeistungT">
  <xsd:_____restriction____ base="xsd:string">
    <xsd:enumeration value="Ue"/>
    <xsd:enumeration value="UeF"/>
    <xsd:enumeration value="HP"/>
    <xsd:enumeration value="VP"/>
    <xsd:enumeration value="AI"/>
    <xsd:enumeration value="AIPlus"/>
  </xsd:_____restriction_____>
</xsd:simpleType>

</xsd:schema>
```

Aufgabe 4:

Die untenstehende Abbildung zeigt die HTML-Ausgabe des Urlaubs-Dokuments aus Aufgabe 1. Die Ausgabe wurde mit dem Stylesheet unten erzeugt.

(a) Vervollständigen Sie es!

(b) Erläutern Sie dann ganz unten, warum der Hotel- bzw. Apartmentname in der Ausgabe erscheint, obwohl dafür kein **`<xsl:value-of select="...">`** angegeben wurde.


```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet version='1.0'
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
    <head><title>Last Minute Angebote</title></head>
    <body>
      <h1>Last Minute ab <xsl:value-of
            select="___Angebote/@Abflug___"/></h1>
      <ul><xsl:apply-templates select="//Angebot" /></ul>
    </body>
  </html>
</xsl:template>

<xsl:template __match="Angebot"__>
  <li><xsl:apply-templates select="Hotel | Apartment"/> in
    <xsl:value-of select="@Ort"/>
    (Region: <xsl:value-of select="@Region"/>)
```

```
    <xsl:apply-templates select="__Preis__"/>
    <xsl:if test="___*__/@Sterne &gt; 3"><b> Komfort! </b></xsl:if>
  </li>
</xsl:template>

<xsl:template match="Preis">
  Preis: <xsl:value-of select="____.____"/> EUR für
      <xsl:value-of select="@Leistung"/>
  <xsl:if test=". &lt; 800"><b> Schnäppchen! </b></xsl:if>
</xsl:template>

</xsl:stylesheet>
```

Hinweis: Beachten Sie, daß Hotels **und** Apartments das Attribut **`Sterne`** haben!

Erläuterung zu (b) - Warum der Hotel- bzw. Apartmentname in der Ausgabe erscheint.

Da kein Template für Hotel- bzw. Apartment-Elemente angegeben ist, wird die Standardvorlage genommen. Diese gibt den Textinhalt, also den Hotel- bzw. Apartmentnamen, aus.

Aufgabe 5:

Die folgende XQuery soll auf dem Urlaubs-Dokument **urlaub.xml** aus Aufgabe 1 laufen, wobei das Dokument dann natürlich wohlgeformt sein muß. Welche Ausgabe wird geliefert?

```
<Komfort>
{ for $a in fn:doc("urlaub.xml")//Angebot
  where $a/*/@Sterne > 3
  return
    <top>
      {fn:string($a/(Hotel | Apartment))} in {fn:string($a/@Ort)}
    </top>
}
</Komfort>
```

```
<Komfort>
  <top>Side Star Park in Side</top>
  <top>Phaedra Beach in Malia</top>
</Komfort>
```

Aufgabe 6:

Gegeben sei die folgende Datenbanktabelle mit Namen **ANGEBOTE**.

REGION	ORT	NAME	TYP	STERNE	PREIS	LEISTUNG
Gran Canaria	Playa del Ingles	Sahara Playa	H	3	974	HP
Gran Canaria	Playa del Ingles	Maritim Playa	H	3	950	HP
Ibiza	Figueretas	Apartamentos Lido	A	3	676	Ue
Antalya	Side	Side Star Park	H	4	974	AI
Kreta	Malia	Phaedra Beach	H	4	928	HP

Vervollständigen Sie die folgende SQL/XML-Abfage so, daß die Ausgabe ganz unten geliefert wird.

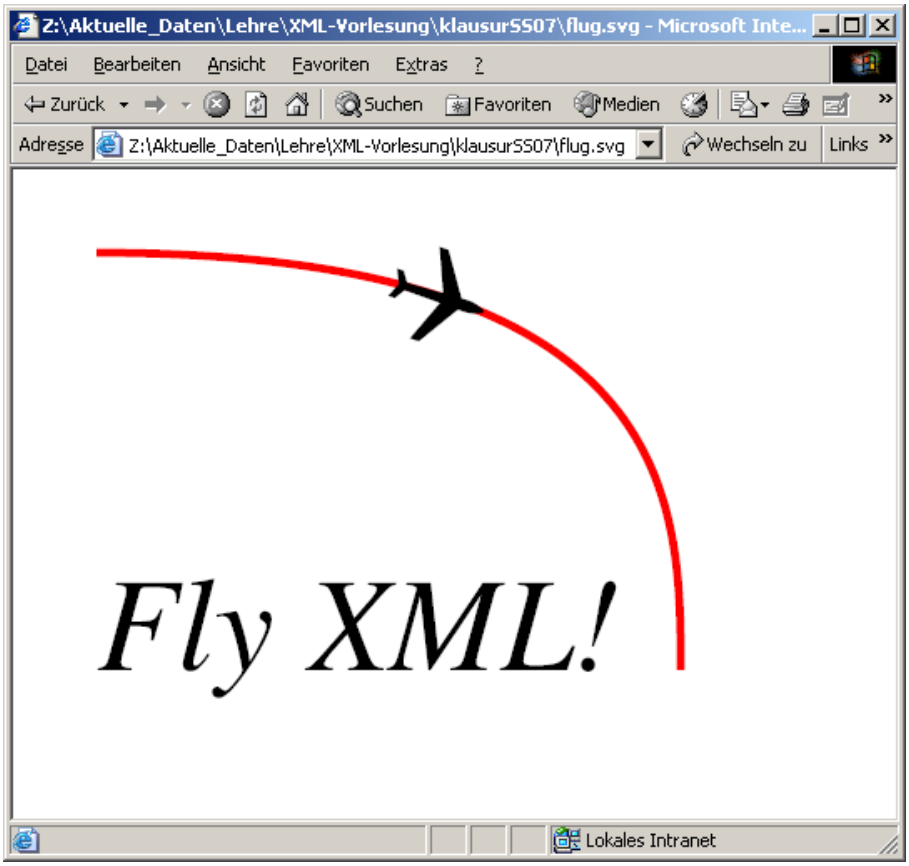
```
SELECT
  XMLELEMENT(NAME "Komfortklasse",
    XMLATTRIBUTES(a.STERNE AS "Sterne"),
    XMLAGG(
    XMLELEMENT(NAME "Angebot",
      XMLATTRIBUTES(a.ORT AS "Ort", a.NAME AS "Name"),
      a.PREIS) ))
FROM ANGEBOTE a
GROUP BY a.STERNE;
```

Ausgabe

```
<Komfortklasse Sterne="3">
  <Angebot Ort="Playa del Ingles" Name="Sahara Playa">974</Angebot>
  <Angebot Ort="Playa del Ingles" Name="Maritim Playa">950</Angebot>
  <Angebot Ort="Figueretas" Name="Apartamentos Lido">676</Angebot>
</Komfortklasse>
<Komfortklasse Sterne="4">
  <Angebot Ort="Side" Name="Side Star Park">974</Angebot>
  <Angebot Ort="Malia" Name="Phaedra Beach">928</Angebot>
</Komfortklasse>
```

Aufgabe 7:

Diese unglaublich elegante, animierte Graphik verwendet die Tatsache, daß es in der Font-Familie Wingdings ein Zeichen gibt, das ein stilisiertes Flugzeug darstellt, das in unserem Fall entlang der Linie fliegt. Für die Lücken im SVG-Dokument geben wir rechts eine Auswahl an. Füllen Sie die Lücken mit den richtigen Nummern!




```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg">
  <path d="M50 50 C 400 50 400 200 400 300" stroke="red"
       stroke-width="____⑦_____" fill="__②__"/>

  <text x="0" y="30" font-family="__⑥__" font-size="80"
       text-anchor="__③____">&#x51;
    <animateMotion path="M50 50 C 400 50 400 200 400 300"
       dur="__⑨__" rotate="___④__" repeatCount="__①___" />
  </text>

  <text x="50" y="300" font-family="Times"
    font-size="__⑧__" font-style="___⑤___">Fly XML!
  </text>
  </svg>
```

① indefinite
② none
③ middle
④ auto
⑤ italic
⑥ Wingdings
⑦ 5
⑧ 80
⑨ 6s

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

Nachname: Vorname:

Matr.Nr.: Studiengang:

Bearbeiten Sie alle Aufgaben! Bei Ankreuzaufgaben können mehrere Antworten richtig sein. Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 120 Minuten.

Aufgabe	max. Punkte	erreichte Punkte
1	2	
2	7	
3	7	
4	5	
5	6	
6	6	
7	7	
Summe	**40**	

Aufgabe 1:

In dieser Klausur geht es um Scheine, d. h. Prüfungsnachweise. Das folgende wohlgeformte und gültige XML-Dokument `Scheinliste.xml` stellt eine vereinfachte Liste von Scheinen dar und ist Grundlage für die folgenden Aufgaben.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Scheinliste SYSTEM "Scheinliste.dtd">

<Scheinliste>
  <Schein Ausstellungsdatum="18.02.2008" Semester="WS0708">
    <Veranstaltung HISNr="4711">Einfuehrung in XML</Veranstaltung>
    <Teilnehmer MatrNr="1234567">A. Mustermann</Teilnehmer>
    <Note Nachweis="Klausur">2,0</Note>
    <Pruefer>L. Wegner</Pruefer>
  </Schein>

  <Schein Ausstellungsdatum="18.02.2008" Semester="WS0708">
    <Veranstaltung HISNr="4711">Einfuehrung in XML</Veranstaltung>
    <Teilnehmer MatrNr="3456789">B. Musterfrau</Teilnehmer>
    <Note Nachweis="Klausur">1,7</Note>
    <Pruefer>L. Wegner</Pruefer>
  </Schein>

  <Schein Ausstellungsdatum="30.07.2007" Semester="SomSem07">
    <Veranstaltung HISNr="4712">Einfuehrung in Tcl/Tk</Veranstaltung>
    <Teilnehmer MatrNr="1234567">A. Mustermann</Teilnehmer>
    <Note Nachweis="MuendlichePruefung">1,3</Note>
    <Pruefer>L. Wegner und K. Schweinsberg</Pruefer>
  </Schein>
</Scheinliste>
```

Wenn zwei oder mehr Attribute in einem Starttag auftreten, wie im Fall von `Ausstellungsdatum` und `Semester` in `<Schein ...>`, dann kann deren **Reihenfolge**

( ) innerhalb aller `Schein`-Starttags beliebig sein.

( ) zwar beliebig sein, muss aber in allen `Schein`-Starttags einheitlich sein.

( ) durch XML-Schema festgelegt werden.

( ) durch eine DTD festgelegt werden.

( ) in DOM-Bäumen durch `position()` abgefragt werden.

Aufgabe 2:

(a) Die folgende DTD beschreibt nicht ganz die Struktur der obigen Liste von Scheinen. Passen Sie die DTD dem Aufbau des XML-Dokuments an!

```
<!-- DTD fuer eine Liste von Scheinen -->

<!ELEMENT Scheinliste (Schein)*>

<!ELEMENT Schein (Note, Veranstaltung, Teilnehmer)>

<!ATTLIST Schein

    Ausstellungsdatum CDATA #REQUIRED

    Semester CDATA #FIXED "WS0708">

<!ELEMENT Note (#PCDATA)>

<!ATTLIST Note

    Nachweis (SchriftlichePruefung | MuendlichePruefung |

          Hausarbeit ) "SchriftlichePruefung">

<!ELEMENT Veranstaltung (#PCDATA)>

<!ATTLIST Veranstaltung HISNr CDATA #IMPLIED>

<!ELEMENT Teilnehmer (#PCDATA)>

<!ATTLIST Teilnehmer MatrNr CDATA #REQUIRED>
```

(b) Welche **Elemente** in einem Schein dürfen weggelassen werden?

(c) Bei welchen **Elementen** ist die Reihenfolge beliebig?

(d) Welches Attribut/welche Attribute dürfen laut Ihrer angepassten DTD weggelassen werden?

Aufgabe 3:

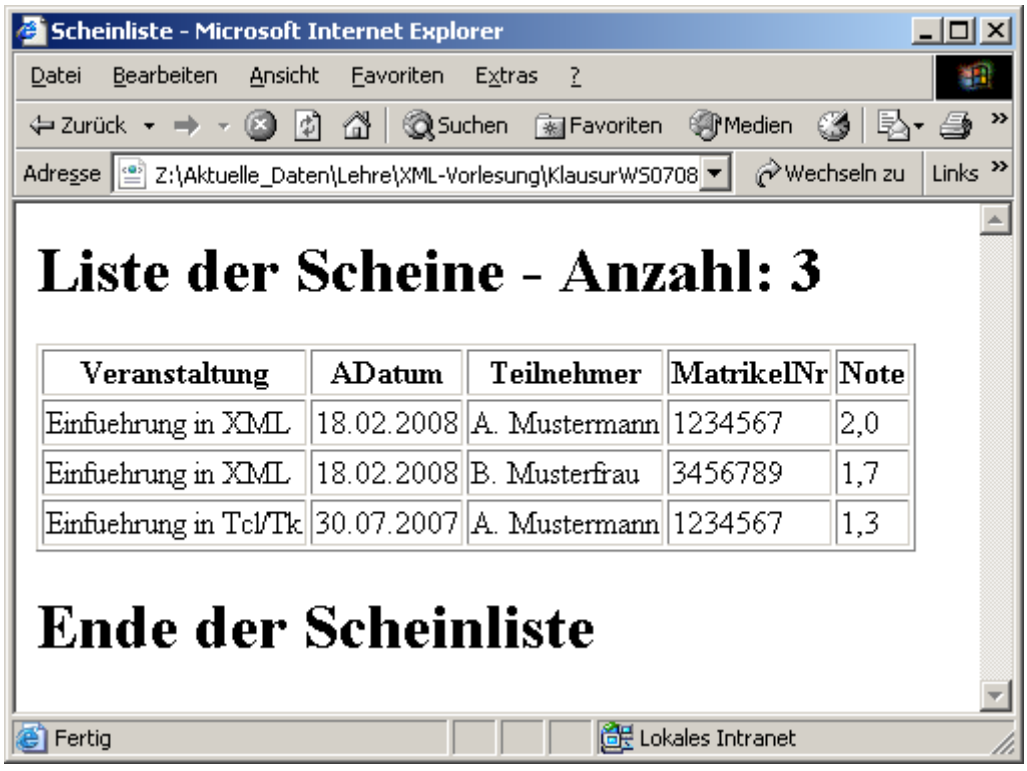
Für die Scheinliste geben wir ein XML Schema an. Ergänzen Sie die fehlenden Teile an den unterstrichenen Stellen. Die Scheinliste kann keine oder unbegrenzt viele Scheine aufnehmen. Die Matrikelnummer `MatrNr` ist eine Pflichtangabe, bei `Nachweis` ist `Klausur` der Defaultwert.

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:element name="Scheinliste" type="ScheinlisteType"/>

  <xsd:complexType name="ScheinlisteType">
    <xsd:sequence>
      <xsd:element name="Schein" type="ScheinType"
          _____________________   ____________________________________ />
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="ScheinType">
    <xsd:sequence>
      <xsd:element name="Veranstaltung" type="_____________________"/>
      <xsd:element name="Teilnehmer" type="_____________________"/>
      <xsd:element name="Note" type="_____________________________"/>
      <xsd:element name="Pruefer" _____________________________/>
    </xsd:sequence>
    <xsd:attribute name="Ausstellungsdatum" type="xsd:string"
        use="required"/>
    <xsd:attribute name="Semester" type="xsd:string" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="VeranstaltungType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute _____________________
            type="xsd:positiveInteger"/>
      </xsd:extension>
  </xsd:simpleContent>
  </xsd:complexType>

  <xsd:complexType name="TeilnehmerType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="MatrNr" type="xsd:positiveInteger"
            ____________________________________ />
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
```

```
  <xsd:complexType name="NoteType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
      <xsd:attribute name="Nachweis" type="NachweisType"
          ____________________________________ />
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>

  <xsd:simpleType name="NachweisType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Klausur"/>
      <xsd:enumeration value="MuendlichePruefung"/>
      <xsd:enumeration value="Hausarbeit"/>
    </xsd:restriction>
  </xsd:simpleType>

</xsd:schema>
```

Aufgabe 4:

Die untenstehende Abbildung zeigt die HTML-Ausgabe einer XML-Scheinliste. Die Ausgabe wurde mit dem Stylesheet unten erzeugt. Vervollständigen Sie das Stylesheet!



```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet version='1.0'
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
    <head>
      <title>Scheinliste </title>
    </head>
    <body>
      <h1>Liste der Scheine - Anzahl:
          <xsl:value-of select="_____________________________________"/>
      </h1>
      <table border="1">
      <tr>
        <th>Veranstaltung</th>
        <th>_____________________________________</th>
        <th>Teilnehmer</th>
        <th>MatrikelNr</th>
        <th>Note</th>
      </tr>
      <_____________________________________________________________/>
      </table>
      <h1>Ende der Scheinliste</h1>
    </body>
  </html>
</xsl:template>

<xsl:template match="Schein">
  <tr>
    <td><xsl:value-of select="Veranstaltung"/></td>
    <td><xsl:value-of select="_____________________________________"/></td>
    <td><xsl:value-of select="Teilnehmer"/></td>
    <td><xsl:value-of select="_____________________________________"/></td>
    <td><xsl:value-of select="Note"/></td>
  </tr>
</xsl:template>

</xsl:stylesheet>
```

Aufgabe 5:

Die folgende XQuery soll aus unserer Scheinliste ein Element `<UnsereBesten>` erzeugen, das - wie in der Ausgabe unten zu sehen - alle Teilnehmer mit einer Note 1,0 oder 1,3 oder 1,7 als `Student`-Elemente enthält. Die Vorlesungsnummer (`HISNr`) und die `Note` werden Attribute, der Name des Teilnehmers wird Inhalt. Ergänzen Sie die Abfrage!

```
<UnsereBesten>

{

    for $s in fn:doc("Scheinliste.xml")//Schein

    where ____________________________________________________________

    return

    <Student ____________________________________________________>

    {

      fn:string(_____________________________________________)

    }

    </Student>

}

</UnsereBesten>
```

Ergebnis:

```
<UnsereBesten>
  <Student LVA="4711" Note="1,7">B. Musterfrau</Student>
  <Student LVA="4712" Note="1,3">A. Mustermann</Student>
</UnsereBesten>
```

Aufgabe 6:

Gegeben sei die folgende verkürzte Datenbanktabelle **ERGEBNISSE**:

VERANSTALTUNG	MATRNR	NOTE
XML	12345678	2
Tcl/Tk	11223344	2
Tcl/Tk	12121212	3
XML	99119911	3
XML	99129912	1
...	...	...
XML	88218821	3
XML	81218121	4
Tcl/Tk	81218131	2
XML	50555055	3
XML	52555255	3

Ergänzen Sie die folgende SQL/XML Abfrage, die einen Notenspiegel nur für die Veranstaltung „XML“ erzeugt. Die gewünschte Ausgabe steht unten.

```
SELECT XMLELEMENT (_____________________________,
  XMLATTRIBUTES(_____________________________),
  COUNT(*))
FROM ERGEBNISSE
WHERE _____________________________________
GROUP BY NOTE
ORDER BY NOTE;
```

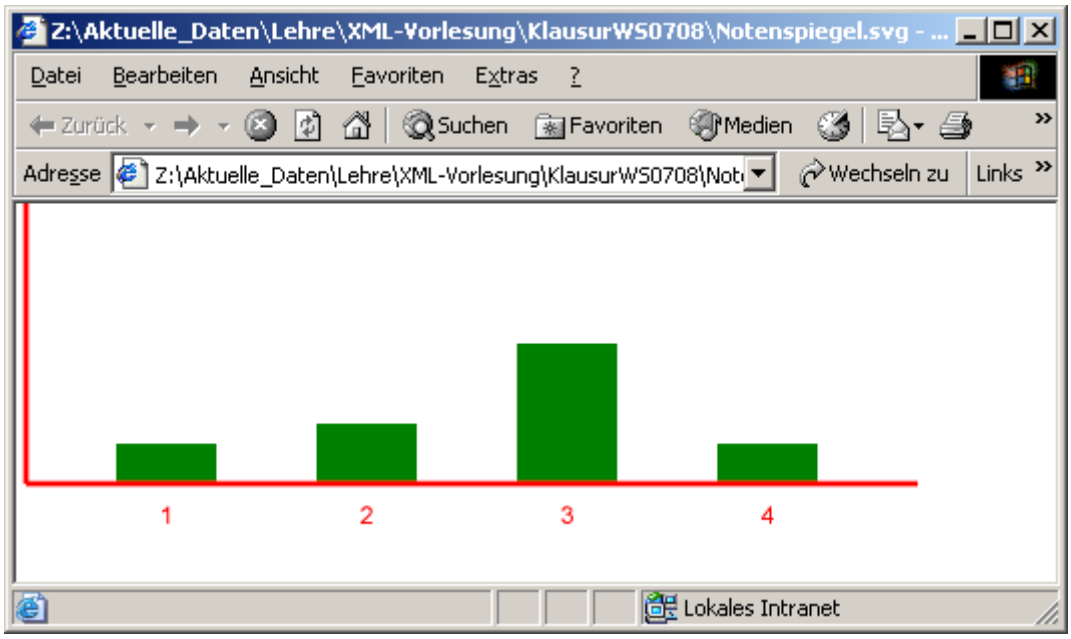
Ausgabe
<Anzahl Note="1">2</Anzahl>
<Anzahl Note="2">3</Anzahl>
<Anzahl Note="3">7</Anzahl>
<Anzahl Note="4">2</Anzahl>

Aufgabe 7:

Es liegt ein XML-Dokument `Notenspiegel.xml` analog zur Ausgabe der Aufgabe 6 vor.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Notenspiegel>
    <Anzahl Note="1">2</Anzahl>
    <Anzahl Note="2">3</Anzahl>
    <Anzahl Note="3">7</Anzahl>
    <Anzahl Note="4">2</Anzahl>
</Notenspiegel>
```

Die untenstehende Grafik dazu wird aus dem SVG-Dokument darunter erzeugt, das wiederum mittels eines Stylesheets generiert wurde. Vervollständigen Sie die Lücken im SVG-Dokument **und** im Stylesheet!




```
<?xml version="1.0" encoding="UTF-8"?>
<svg xmlns="http://www.w3.org/2000/svg">
<rect x="50" y="120" width="50" height="20" fill="green"/>
<text x="75" y="160" text-anchor="middle" fill="red">1</text>
<rect x="150" y="110" width="50" height="30" fill="green"/>
<text x="175" y="160" text-anchor="middle" fill="red">2</text>
<rect x="250" y="70" width="50" height="70" fill="green"/>
<text x="275" y="160" text-anchor="middle" fill="red">3</text>
<rect x="350" y="120" width="50" height="20" fill="green"/>
<text x="375" y="160" text-anchor="middle" fill="red">4</text>
<line x1="___" y1="___" x2="___" y2="___" stroke="red" stroke-width="3"/>
<line x1="___" y1="___" x2="___" y2="___" stroke="red" stroke-width="3"/>
</svg>
```

Das zugehörige Stylesheet:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns="http://www.w3.org/2000/svg">

<xsl:variable name="AnzahlStufen" select="count(//Anzahl)"/>
<xsl:variable name="MaxPersonen" select="sum(//Anzahl)"/>

<xsl:template match="/">
  <svg>
    <xsl:apply-templates select="Notenspiegel/Anzahl"/>
    <line x1="5" y1="0" x2="5" y2="{$MaxPersonen*10}"
        stroke="red" stroke-width="3"/>
    <line x1="5" y1="{$MaxPersonen*10}"
        x2="{$AnzahlStufen*100 + 50}" y2="{$MaxPersonen*10}"
        stroke="red" stroke-width="3"/>
  </svg>
</xsl:template>

<xsl:template ____________________________________ >
  <rect x="{(position()-1)*100 + 50}" y="{$MaxPersonen*10 - current()*10}"
      width="50" height="{current()*10}" ____________________________________ />
  <text x="{(position()-1)*100 + 75}" y="{$MaxPersonen*10 + 20}"
      _____________________________   ____________________________________ >

    <xsl:value-of select="_____________________"/>
  </text>
</xsl:template>

</xsl:stylesheet>
```

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

Nachname: MUSTERLÖSUNG Vorname:

Matr.Nr.: Studiengang:

Bearbeiten Sie alle Aufgaben! Bei Ankreuzaufgaben können mehrere Antworten richtig sein. Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 120 Minuten.

Aufgabe	max. Punkte	erreichte Punkte
1	2	
2	7	
3	7	
4	5	
5	6	
6	6	
7	7	
Summe	**40**	

Aufgabe 1:

In dieser Klausur geht es um Scheine, d. h. Prüfungsnachweise. Das folgende wohlgeformte und gültige XML-Dokument `Scheinliste.xml` stellt eine vereinfachte Liste von Scheinen dar und ist Grundlage für die folgenden Aufgaben.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Scheinliste SYSTEM "Scheinliste.dtd">

<Scheinliste>
  <Schein Ausstellungsdatum="18.02.2008" Semester="WS0708">
    <Veranstaltung HISNr="4711">Einfuehrung in XML</Veranstaltung>
    <Teilnehmer MatrNr="1234567">A. Mustermann</Teilnehmer>
    <Note Nachweis="Klausur">2,0</Note>
    <Pruefer>L. Wegner</Pruefer>
  </Schein>

  <Schein Ausstellungsdatum="18.02.2008" Semester="WS0708">
    <Veranstaltung HISNr="4711">Einfuehrung in XML</Veranstaltung>
    <Teilnehmer MatrNr="3456789">B. Musterfrau</Teilnehmer>
    <Note Nachweis="Klausur">1,7</Note>
    <Pruefer>L. Wegner</Pruefer>
  </Schein>

  <Schein Ausstellungsdatum="30.07.2007" Semester="SomSem07">
    <Veranstaltung HISNr="4712">Einfuehrung in Tcl/Tk</Veranstaltung>
    <Teilnehmer MatrNr="1234567">A. Mustermann</Teilnehmer>
    <Note Nachweis="MuendlichePruefung">1,3</Note>
    <Pruefer>L. Wegner und K. Schweinsberg</Pruefer>
  </Schein>
</Scheinliste>
```

Wenn zwei oder mehr Attribute in einem Starttag auftreten, wie im Fall von `Ausstellungsdatum` und `Semester` in <`Schein` ...>, dann kann deren **Reihenfolge**

(X)innerhalb aller `Schein`-Starttags beliebig sein.

( ) zwar beliebig sein, muss aber in allen `Schein`-Starttags einheitlich sein.

( ) durch XML-Schema festgelegt werden.

( ) durch eine DTD festgelegt werden.

( ) in DOM-Bäumen durch `position()` abgefragt werden.

Aufgabe 2:

(a) Die folgende DTD beschreibt nicht ganz die Struktur der obigen Liste von Scheinen. Passen Sie die DTD dem Aufbau des XML-Dokuments an!

```
<!-- DTD fuer eine Liste von Scheinen -->

<!ELEMENT Scheinliste (Schein)*>
                          Veranstaltung, Teilnehmer, Note, Pruefer
<!ELEMENT Schein (Note, Veranstaltung, Teilnehmer)>

<!ATTLIST Schein

    Ausstellungsdatum CDATA #REQUIRED

    Semester CDATA #FIXED "WS0708">  #REQUIRED oder #IMPLIED

<!ELEMENT Note (#PCDATA)>

<!ATTLIST Note
                    Klausur
    Nachweis (SchriftlichePruefung | MuendlichePruefung |

          Hausarbeit ) "SchriftlichePruefung">
                                    Klausur
<!ELEMENT Veranstaltung (#PCDATA)>

<!ATTLIST Veranstaltung HISNr CDATA #IMPLIED>

<!ELEMENT Teilnehmer (#PCDATA)>

<!ATTLIST Teilnehmer MatrNr CDATA #REQUIRED>

<!ELEMENT Pruefer (#PCDATA)>
```

(b) Welche **Elemente** in einem Schein dürfen weggelassen werden?

keine

(c) Bei welchen **Elementen** ist die Reihenfolge beliebig?

bei keinen

(d) Welches Attribut/welche Attribute dürfen laut Ihrer angepassten DTD weggelassen werden?

HISNr, Nachweis, ggf. Semester wenn #IMPLIED

Aufgabe 3:

Für die Scheinliste geben wir ein XML Schema an. Ergänzen Sie die fehlenden Teile an den unterstrichenen Stellen. Die Scheinliste kann keine oder unbegrenzt viele Scheine aufnehmen. Die Matrikelnummer `MatrNr` ist eine Pflichtangabe, bei `Nachweis` ist `Klausur` der Defaultwert.

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:element name="Scheinliste" type="ScheinlisteType"/>

  <xsd:complexType name="ScheinlisteType">
    <xsd:sequence>
      <xsd:element name="Schein" type="ScheinType"
          __minOccurs="0"__   ___maxOccurs="unbounded"___ />
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="ScheinType">
    <xsd:sequence>
      <xsd:element name="Veranstaltung" type="__VeranstaltungType__"/>
      <xsd:element name="Teilnehmer" type="__TeilnehmerType__"/>
      <xsd:element name="Note" type="__NoteType__"/>
      <xsd:element name="Pruefer" __type="xsd:string"__/>
    </xsd:sequence>
    <xsd:attribute name="Ausstellungsdatum" type="xsd:string"
        use="required"/>
    <xsd:attribute name="Semester" type="xsd:string" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="VeranstaltungType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute __name="HISNr"__
            type="xsd:positiveInteger"/>
      </xsd:extension>
  </xsd:simpleContent>
  </xsd:complexType>

  <xsd:complexType name="TeilnehmerType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="MatrNr" type="xsd:positiveInteger"
            __use="required"__ />
      </xsd:extension>
    </xsd:simpleContent>
```

```
  </xsd:complexType>

  <xsd:complexType name="NoteType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
      <xsd:attribute name="Nachweis" type="NachweisType"
          __default="Klausur"__ />
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>

  <xsd:simpleType name="NachweisType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Klausur"/>
      <xsd:enumeration value="MuendlichePruefung"/>
      <xsd:enumeration value="Hausarbeit"/>
    </xsd:restriction>
  </xsd:simpleType>

</xsd:schema>
```

Aufgabe 4:

Die untenstehende Abbildung zeigt die HTML-Ausgabe einer XML-Scheinliste. Die Ausgabe wurde mit dem Stylesheet unten erzeugt. Vervollständigen Sie das Stylesheet!

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet version='1.0'
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
    <head>
      <title>Scheinliste </title>
    </head>
    <body>
      <h1>Liste der Scheine - Anzahl:
          <xsl:value-of select="__count(//Schein)__"/>
      </h1>
      <table border="1">
      <tr>
        <th>Veranstaltung</th>
        <th>__ADatum__</th>
        <th>Teilnehmer</th>
        <th>MatrikelNr</th>
        <th>Note</th>
      </tr>
      <_xsl:apply-templates select="//Schein"_/>     auch richtig ohne select=...
      </table>
      <h1>Ende der Scheinliste</h1>
    </body>
  </html>
</xsl:template>

<xsl:template match="Schein">
  <tr>
    <td><xsl:value-of select="Veranstaltung"/></td>
    <td><xsl:value-of select="__@Ausstellungsdatum__"/></td>
    <td><xsl:value-of select="Teilnehmer"/></td>
    <td><xsl:value-of select="__Teilnehmer/@MatrNr__"/></td>
    <td><xsl:value-of select="Note"/></td>
  </tr>
</xsl:template>

</xsl:stylesheet>
```

Aufgabe 5:

Die folgende XQuery soll aus unserer Scheinliste ein Element `<UnsereBesten>` erzeugen, das - wie in der Ausgabe unten zu sehen - alle Teilnehmer mit einer Note 1,0 oder 1,3 oder 1,7 als `Student`-Elemente enthält. Die Vorlesungsnummer (`HISNr`) und die `Note` werden Attribute, der Name des Teilnehmers wird Inhalt. Ergänzen Sie die Abfrage!

```
<UnsereBesten>

{

    for $s in fn:doc("Scheinliste.xml")//Schein

    where __$s/Note="1,0" or $s/Note="1,3" or $s/Note="1,7"__

    return

    <Student _LVA="{$s/Veranstaltung/@HisNr}" Note="{$s/Note}"_>

    {

      fn:string(__$s/Teilnehmer__)

    }

    </Student>

}

</UnsereBesten>
```

Ergebnis:

```
<UnsereBesten>
  <Student LVA="4711" Note="1,7">B. Musterfrau</Student>
  <Student LVA="4712" Note="1,3">A. Mustermann</Student>
</UnsereBesten>
```

Aufgabe 6:

Gegeben sei die folgende verkürzte Datenbanktabelle **ERGEBNISSE**:

VERANSTALTUNG	MATRNR	NOTE
XML	12345678	2
Tcl/Tk	11223344	2
Tcl/Tk	12121212	3
XML	99119911	3
XML	99129912	1
...	...	...
XML	88218821	3
XML	81218121	4
Tcl/Tk	81218131	2
XML	50555055	3
XML	52555255	3

Ergänzen Sie die folgende SQL/XML Abfrage, die einen Notenspiegel nur für die Veranstaltung „XML“ erzeugt. Die gewünschte Ausgabe steht unten.

```
SELECT XMLELEMENT (_NAME "Anzahl"___,
  XMLATTRIBUTES(__NOTE AS "Note"__),
  COUNT(*))
FROM ERGEBNISSE
WHERE __VERANSTALTUNG = 'XML'__
GROUP BY NOTE
ORDER BY NOTE;
```

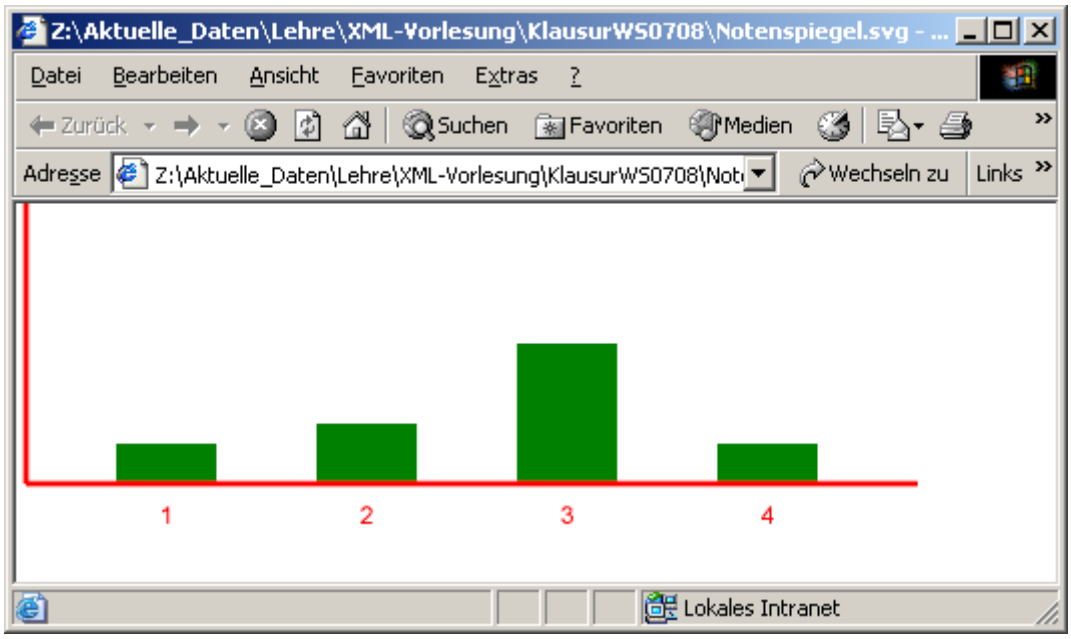
Ausgabe
<Anzahl Note="1">2</Anzahl>
<Anzahl Note="2">3</Anzahl>
<Anzahl Note="3">7</Anzahl>
<Anzahl Note="4">2</Anzahl>

Aufgabe 7:

Es liegt ein XML-Dokument `Notenspiegel.xml` analog zur Ausgabe der Aufgabe 6 vor.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Notenspiegel>
    <Anzahl Note="1">2</Anzahl>
    <Anzahl Note="2">3</Anzahl>
    <Anzahl Note="3">7</Anzahl>
    <Anzahl Note="4">2</Anzahl>
</Notenspiegel>
```

Die untenstehende Grafik dazu wird aus dem SVG-Dokument darunter erzeugt, das wiederum mittels eines Stylesheets generiert wurde. Vervollständigen Sie die Lücken im SVG-Dokument **und** im Stylesheet!




```
<?xml version="1.0" encoding="UTF-8"?>
<svg xmlns="http://www.w3.org/2000/svg">
<rect x="50" y="120" width="50" height="20" fill="green"/>
<text x="75" y="160" text-anchor="middle" fill="red">1</text>
<rect x="150" y="110" width="50" height="30" fill="green"/>
<text x="175" y="160" text-anchor="middle" fill="red">2</text>
<rect x="250" y="70" width="50" height="70" fill="green"/>
<text x="275" y="160" text-anchor="middle" fill="red">3</text>
<rect x="350" y="120" width="50" height="20" fill="green"/>
<text x="375" y="160" text-anchor="middle" fill="red">4</text>
<line x1="5" y1="0" x2="5" y2="140" stroke="red" stroke-width="3"/>
<line x1="5" y1="140" x2="450" y2="140" stroke="red" stroke-width="3"/>
</svg>
```

Das zugehörige Stylesheet:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns="http://www.w3.org/2000/svg">

<xsl:variable name="AnzahlStufen" select="count(//Anzahl)"/>
<xsl:variable name="MaxPersonen" select="sum(//Anzahl)"/>

<xsl:template match="/">
  <svg>
    <xsl:apply-templates select="Notenspiegel/Anzahl"/>
    <line x1="5" y1="0" x2="5" y2="{$MaxPersonen*10}"
        stroke="red" stroke-width="3"/>
    <line x1="5" y1="{$MaxPersonen*10}"
        x2="{$AnzahlStufen*100 + 50}" y2="{$MaxPersonen*10}"
        stroke="red" stroke-width="3"/>
  </svg>
</xsl:template>

<xsl:template __match="Anzahl"__ >
  <rect x="{(position()-1)*100 + 50}" y="{$MaxPersonen*10 - current()*10}"
      width="50" height="{current()*10}" __fill="green"___ />
  <text x="{(position()-1)*100 + 75}" y="{$MaxPersonen*10 + 20}"
      _text-anchor="middle"__   __fill="red"__ >

    <xsl:value-of select="__@Note__"/>
  </text>
</xsl:template>

</xsl:stylesheet>
```

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

Nachname: Vorname:

Matr.Nr.: Studiengang:

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 90 Minuten.

Aufgabe	Punkte max.	Punkte erreicht
1	3+2	
2	4+2+2+2	
3	8	
4	5+3	
5	6	
6	6	
7	6	
Summe	**49**	

Aufgabe 1:

Das folgende Dokument **kameras.xml** stellt vereinfacht eine Liste gängiger Kameras dar[1].

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Angebote SYSTEM "kameras.dtd">
<Angebote Kategorie="Digitale SLR" vom="10.07.2008">

  <Angebot Aid="C4711" Marke="Canon" Preis="398,50">
    <Modell Megapixel="10.1">EOS 400D Kit mit EF S 18-55</Modell>
    <Bemerkung>meistverkaufte digitale Spiegelreflex</Bemerkung>
  </Angebot>
  <Angebot Aid="C4712" Marke="Canon" Preis="599,50">
    <Modell Megapixel="12.2">EOS 450D Kit mit EF S 18-55</Modell>
    <Bemerkung>mit Live-View</Bemerkung>
  </Angebot>
  <Angebot Aid="N2927" Marke="Nikon" Preis="378,00">
    <Modell Megapixel="6.1">D40 Kit mit AF-S DX Nikkor 18-55</Modell>
    <Bemerkung>Modell schwarz, handliche Reisekamera</Bemerkung>
  </Angebot>
  <Angebot Aid="N2423" Marke="Nikon" Preis="358,00">
    <Modell Megapixel="6.1">D40 Kit mit AF-S DX Nikkor 18-55</Modell>
    <Bemerkung>Body und Objektiv silber, Restexemplar neu</Bemerkung>
  </Angebot>
  <Angebot Aid="P0965" Marke="Pentax" Preis="529,00">
    <Modell Megapixel="10.2">K 200 D Kit mit DA 18-55 AL II</Modell>
    <Bemerkung>Bildstabil., staub- und wassergeschuetzt</Bemerkung>
  </Angebot>

</Angebote>
```

(a) Ist das Dokument wohlgeformt? Wenn nein, markieren Sie Fehler deutlich.

(b) Enthält das Dokument ein leeres Element? Wenn ja, geben Sie es an!

(c) Hat das Dokument ein Wurzelelement? Wenn ja, geben Sie es an!

1. Die Auswahl der Modelle ist willkürlich und stellt in keiner Weise eine Bewertung dar.

Aufgabe 2:

(a) Hier ist die DTD für das Kamera-Dokument aus Aufgabe 1.

```
<!-- DTD fuer Kameraangebote -->
<!ELEMENT Angebote (Angebot)+>
  <!ATTLIST Angebote Kategorie CDATA #REQUIRED
    vom CDATA #REQUIRED>

<!ELEMENT Angebot (Modell, Bemerkung?)>
  <!ATTLIST Angebot Aid ID #REQUIRED
    Preis CDATA #REQUIRED
    Marke (Canon | Fuji | Leica | Nikon | Olympus | Pentax | Sony |
        NB) "NB">

<!ELEMENT Modell (#PCDATA)>
  <!ATTLIST Modell Megapixel CDATA #IMPLIED>

<!ELEMENT Bemerkung (#PCDATA)>
```

(a) Ist das Dokument von Aufgabe 1 gültig (valid) bezüglich dieser DTD?

(b) Wenn wie im Fall des Elements **`Angebot`** die beiden Unterelemente **`Modell`** und **`Bemerkung`** in dieser Reihenfolge in der DTD angegeben werden, dann wäre eine Reihenfolge **`Bemerkung Modell`** im Dokument verboten. Ist diese Aussage richtig oder falsch?

(c) Das Attribute **`Megapixel`** sollte genau eine Nachkommastelle haben. Wie geht das mit einer DTD?

(d) Wäre es möglich, zwei Angeboten die selbe Angebot-ID **`Aid`** zuzuweisen? Begründung!

(e) Wäre es möglich, im Dokument ein Angebot für eine Kamera mit dem Markenattributwert „Kodak“ aufzunehmen? Begründung!

Aufgabe 3:

Ein XML-Schema zum Kamera-Dokument sieht wie folgt aus. Füllen Sie die Lücken aus.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Angebote" type="AngeboteType"/>

  <xsd:complexType name="AngeboteType">
    <xsd:sequence>
      <xsd:element name="Angebot" type="AngebotType"
          minOccurs="1" maxOccurs="_____________________"/>
    </xsd:sequence>
    <xsd:attribute name="Kategorie" type="xsd:string"
          use="required"/>
    <xsd:attribute name="_______" type="xsd:string" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="AngebotType">
    <xsd:sequence>
      <xsd:element name="Modell" type="ModellType"/>
      <xsd:element name="Bemerkung" type="xsd:string"
          minOccurs="_________________" maxOccurs="_________________"/>
    </xsd:sequence>
    <xsd:attribute name="Aid" type="xsd:ID" use="required"/>
    <xsd:attribute name="Preis" type="xsd:string" use="required"/>
    <xsd:attribute name="Marke" type="MarkeType" _____________="NB"/>
  </xsd:complexType>

  <xsd:complexType name="ModellType">
    <xsd:_____________________________________>
      <xsd:extension base="_____________________">
        <xsd:attribute name="Megapixel" type="xsd:decimal"/>
      </xsd:extension>
    </xsd:_____________________________________>
  </xsd:complexType>

  <xsd:simpleType name="_____________________________">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Canon"/>
```

```
        <xsd:enumeration value="Fuji"/>
        <xsd:enumeration value="Leica"/>
        <xsd:enumeration value="Nikon"/>
        <xsd:enumeration value="Olympus"/>
        <xsd:enumeration value="Pentax"/>
        <xsd:enumeration value="Sony"/>
        <xsd:enumeration value="NB"/>
      </xsd:restriction>
    </xsd:simpleType>

</xsd:schema>
```

Aufgabe 4:

Die untenstehende Abbildung zeigt die HTML-Ausgabe des Kamera-Dokuments aus Aufgabe 1. Die Ausgabe wurde mit dem Stylesheet unten erzeugt.

Vervollständigen Sie es!

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<xsl:stylesheet version='1.0'
  xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
  <xsl:template match='_______________'>
    <HTML>
      <HEAD><TITLE>Kameras</TITLE></HEAD>
      <__________________>
        <H1 align='center'>
          Kameraliste für die Kategorie
          <xsl:value-of select='Angebote/@Kategorie'/>
          vom
          <xsl:value-of select='_____________________________'/>
        </H1>
        <xsl:apply-templates/>
      </BODY>
    </HTML>
  </xsl:template>
  <xsl:__________________ _______________='Angebote'>
    <TABLE border='1' align='center' cellpadding='5'>
      <TR>
        <TH>Foto</TH>
        <TH>Marke</TH>
        <TH>Modell</TH>
        <TH>Megapixel</TH>
        <TH>Bemerkung</TH>
        <TH>Preis</TH>
      </TR>
      <xsl:apply-templates/>
    </TABLE>
  </xsl:_____________________>
  <xsl:template match='_____________________________'>
    <TR>
      <TD><IMG SRC="{concat(@Aid,'.jpg')}"/></TD>
      <TD><xsl:value-of select='@Marke'/></TD>
      <TD><xsl:value-of select='Modell'/></TD>
      <TD><xsl:value-of select='_____________________________'/></TD>
      <TD><xsl:value-of select='Bemerkung'/></TD>
      <TD><xsl:value-of select='_____________________________'/></TD>
    </TR>
  </xsl:template>
</xsl:stylesheet>
```

Aufgabe 5:

Die folgende XQuery soll auf dem Kamera-Dokument **`kameras.xml`** aus Aufgabe 1 laufen, wobei das Dokument dann natürlich wohlgeformt sein muß. Welche Ausgabe wird geliefert?

```
<VielePixel>
{
  for $a in fn:doc("kameras.xml")//Angebot
    where $a/Modell/@Megapixel > 8
    order by $a/@Preis
    return
      <Kamera Aid="{ $a/@Aid }" Preis="{ $a/@Preis }">
        {
          fn:string($a/Modell/@Megapixel)
        }
      </Kamera>
}
</VielePixel>
```

Aufgabe 6:

Gegeben sei die folgende Datenbanktabelle mit Namen **ANGEBOTE**.

AID	MARKE	MODELL	MPIXEL	BEMERKUNG	PREIS
C4711	Canon	EOS 400D Kit mit EF S 18-55	10.1	meistverkaufte digitale Spiegelreflex	398,50
C4712	Canon	EOS 450D Kit mit EF S 18-55	12.2	mit Live-View	599,50
N2927	Nikon	D40 Kit mit AF-S DX Nikkor 18-55	6.1	Modell schwarz, handliche Reisekamera	378,00
N2423	Nikon	D40 Kit mit AF-S DX Nikkor 18-55	6.1	Body und Objektiv silber, Restexemplar neu	358,00
P0965	Pentax	K 200 D Kit mit DA 18-55 AL II	10.2	Bildstabil., staub- und wassergeschuetzt	529,00

Vervollständigen Sie die folgende SQL/XML-Abfage so, daß das Dokument aus Aufgabe 1 geliefert wird. Dabei wird allerdings das Wurzelelement **`Angebote`** nicht produziert.

```
SELECT
  ____________________________________(
    NAME "Angebot",
    ____________________________________(
      AID AS "Aid",
      MARKE AS "Marke",
      PREIS AS "Preis"
    ),
    ____________________________________(
      NAME "Modell",
      ____________________________________(
        MPIXEL AS "Megapixel"
      ),
      MODELL
    ),
    ____________________________________(
      NAME "Bemerkung",
      BEMERKUNG
    )
  )
FROM ____________________________________________;
```

Aufgabe 7:

Die Firma Foto trübe-linse GmbH&Co KG hat zwar kein so tolles Angebot an Kameras, dafür aber ein animiertes Logo. Das SVG-Dokument verwendet ein Zeichen aus der Font-Familie Webdings, das eine stilisierte Kamera darstellt, das in unserem Fall in zwei Exemplaren entlang der Linie fliegt. Für die Lücken im SVG-Dokument geben wir rechts eine Auswahl an. Füllen Sie die Lücken mit den richtigen Nummern, wobei einige Begriffe garnicht, andere mehrfach passen können!

① **repeatCount**
② **dur**
③ **Foto**
④ **symbol**
⑤ **path**
⑥ **font-family**
⑦ **fill**
⑧ **canvas**
⑨ **stroke**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<svg xmlns="http://www.w3.org/2000/svg">
  <_____ d="M500,100 A100,50 0 1,1 500,500 100,50 0 1,1 500,100" _______="red"
      stroke-width="5" ________="none"/>
  <text x="0" y="15" _____________________="Webdings" font-size="100"
      text-anchor="middle">&#xB5;
    <animateMotion path="M500,100 A100,50 0 1,1 500,500 100,50 0 1,1 500,100"
        _____="30s" rotate="auto" _____________________="indefinite" />
  </text>
  <text x="0" y="15" _____________________="Webdings" font-size="100"
      text-anchor="middle">&#xB5;
    <animateMotion path="M500,500 A100,50 0 1,1 500,100 100,50 0 1,1 500,500"
        _____="30s" rotate="auto" _____________________="indefinite" />
  </text>
  <text x="500" y="200" font-family="Times" font-size="60"
      text-anchor="middle" font-style="italic">_____________</text>
  <text x="500" y="310" font-family="Times" font-size="100"
      text-anchor="middle" font-style="italic">t r ü b e - l i n s e</text>
  <text x="500" y="420" font-family="Times" font-size="60"
      text-anchor="middle" font-style="italic">GmbH &amp;amp; Co. KG</text>
</svg>
```

Klausur zur Vorlesung „Einführung in XML“

MUSTERLÖSUNG

Nachname: Vorname:

Matr.Nr.: Studiengang:

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 90 Minuten.

Aufgabe	Punkte max.	Punkte erreicht
1	2+1+1	
2	2+1+1+1+1	
3	8	
4	8	
5	6	
6	6	
7	5	
Summe	**43**	

Aufgabe 1:

Das folgende Dokument **kameras.xml** stellt vereinfacht eine Liste gängiger Kameras dar[1].

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Angebote SYSTEM "kameras.dtd">
<Angebote Kategorie="Digitale SLR" vom="10.07.2008">

  <Angebot Aid="C4711" Marke="Canon" Preis="398,50">
    <Modell Megapixel="10.1">EOS 400D Kit mit EF S 18-55</Modell>
    <Bemerkung>meistverkaufte digitale Spiegelreflex</Bemerkung>
  </Angebot>
  <Angebot Aid="C4712" Marke="Canon" Preis="599,50">
    <Modell Megapixel="12.2">EOS 450D Kit mit EF S 18-55</Modell>
    <Bemerkung>mit Live-View</Bemerkung>
  </Angebot>
  <Angebot Aid="N2927" Marke="Nikon" Preis="378,00">
    <Modell Megapixel="6.1">D40 Kit mit AF-S DX Nikkor 18-55</Modell>
    <Bemerkung>Modell schwarz, handliche Reisekamera</Bemerkung>
  </Angebot>
  <Angebot Aid="N2423" Marke="Nikon" Preis="358,00">
    <Modell Megapixel="6.1">D40 Kit mit AF-S DX Nikkor 18-55</Modell>
    <Bemerkung>Body und Objektiv silber, Restexemplar neu</Bemerkung>
  </Angebot>
  <Angebot Aid="P0965" Marke="Pentax" Preis="529,00">
    <Modell Megapixel="10.2">K 200 D Kit mit DA 18-55 AL II</Modell>
    <Bemerkung>Bildstabil., staub- und wassergeschuetzt</Bemerkung>
  </Angebot>

</Angebote>
```

(a) Ist das Dokument wohlgeformt? Wenn nein, markieren Sie Fehler deutlich.

Ja

(b) Enthält das Dokument ein leeres Element? Wenn ja, geben Sie es an!

Nein

(c) Hat das Dokument ein Wurzelelement? Wenn ja, geben Sie es an!

Ja, „Angebote"

1. Die Auswahl der Modelle ist willkürlich und stellt in keiner Weise eine Bewertung dar.

Aufgabe 2:

(a) Hier ist die DTD für das Kamera-Dokument aus Aufgabe 1.

```
<!-- DTD fuer Kameraangebote -->
<!ELEMENT Angebote (Angebot)+>
<!ATTLIST Angebote Kategorie CDATA #REQUIRED
                   vom CDATA #REQUIRED>

<!ELEMENT Angebot (Modell, Bemerkung?)>
<!ATTLIST Angebot Aid ID #REQUIRED
                  Preis CDATA #REQUIRED
                  Marke (Canon | Fuji | Leica | Nikon | Olympus |
                         Pentax | Sony | NB) "NB">

<!ELEMENT Modell (#PCDATA)>
<!ATTLIST Modell Megapixel CDATA #IMPLIED>

<!ELEMENT Bemerkung (#PCDATA)>
```

(a) Ist das Dokument von Aufgabe 1 gültig (valid) bezüglich dieser DTD?

Ja

(b) Dürfen nach dieser DTD im XML-Dokument die beiden Unterelemente **`Modell`** und **`Bemerkung`** in der Reihenfolge vertauscht werden?

Nein

(c) Das Attribute **`Megapixel`** sollte genau eine Nachkommastelle haben. Wie geht das mit einer DTD?

Garnicht

(d) Wäre es möglich, zwei Angeboten die selbe Angebot-ID **`Aid`** zuzuweisen? Begründung!

Nein, weil vom Typ ID

(e) Wäre es möglich, im Dokument ein Angebot für eine Kamera mit dem Markenattribut-wert „Kodak“ aufzunehmen? Begründung!

Nein, weil nicht in der Aufzählung enthalten

Aufgabe 3:

Ein XML-Schema zum Kamera-Dokument sieht wie folgt aus. Füllen Sie die Lücken aus.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Angebote" type="AngeboteType"/>

  <xsd:complexType name="AngeboteType">
    <xsd:sequence>
      <xsd:element name="Angebot" type="AngebotType"
          minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="Kategorie" type="xsd:string"
          use="required"/>
    <xsd:attribute name="vom" type="xsd:string" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="AngebotType">
    <xsd:sequence>
      <xsd:element name="Modell" type="ModellType"/>
      <xsd:element name="Bemerkung" type="xsd:string"
          minOccurs="_______0________" maxOccurs="_______1_______"/>
    </xsd:sequence>
    <xsd:attribute name="Aid" type="xsd:ID" use="required"/>
    <xsd:attribute name="Preis" type="xsd:string" use="required"/>
    <xsd:attribute name="Marke" type="MarkeType" default="NB"/>
  </xsd:complexType>

  <xsd:complexType name="ModellType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="Megapixel" type="xsd:decimal"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>

  <xsd:simpleType name="MarkeType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Canon"/>
```

```
        <xsd:enumeration value="Fuji"/>
        <xsd:enumeration value="Leica"/>
        <xsd:enumeration value="Nikon"/>
        <xsd:enumeration value="Olympus"/>
        <xsd:enumeration value="Pentax"/>
        <xsd:enumeration value="Sony"/>
        <xsd:enumeration value="NB"/>
      </xsd:restriction>
    </xsd:simpleType>

</xsd:schema>
```

Aufgabe 4:

Die untenstehende Abbildung zeigt die HTML-Ausgabe des Kamera-Dokuments aus Aufgabe 1. Die Ausgabe wurde mit dem Stylesheet unten erzeugt.

Vervollständigen Sie es!

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<xsl:stylesheet version='1.0'
  xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
  <xsl:template match='_____/_____'>
    <HTML>
      <HEAD><TITLE>Kameras</TITLE></HEAD>
      <BODY>
        <H1 align='center'>
          Kameraliste für die Kategorie
          <xsl:value-of select='Angebote/@Kategorie'/>
          vom
          <xsl:value-of select='Angebote/@vom'/>
        </H1>
        <xsl:apply-templates/>
      </BODY>
    </HTML>
  </xsl:template>
  <xsl:template match='Angebote'>
    <TABLE border='1' align='center' cellpadding='5'>
      <TR>
        <TH>Foto</TH>
        <TH>Marke</TH>
        <TH>Modell</TH>
        <TH>Megapixel</TH>
        <TH>Bemerkung</TH>
        <TH>Preis</TH>
      </TR>
      <xsl:apply-templates/>
    </TABLE>
  </xsl:template>
  <xsl:template match='Angebot'>
    <TR>
      <TD><IMG SRC="{concat(@Aid,'.jpg')}"/></TD>
      <TD><xsl:value-of select='@Marke'/></TD>
      <TD><xsl:value-of select='Modell'/></TD>
      <TD><xsl:value-of select='Modell/@Megapixel'/></TD>
      <TD><xsl:value-of select='Bemerkung'/></TD>
      <TD><xsl:value-of select='@Preis'/></TD>
    </TR>
  </xsl:template>
</xsl:stylesheet>
```

Alternativ: Angebote/Angebot

Aufgabe 5:

Die folgende XQuery soll auf dem Kamera-Dokument **`kameras.xml`** aus Aufgabe 1 laufen, wobei das Dokument dann natürlich wohlgeformt sein muß. Welche Ausgabe wird geliefert?

```
<VielePixel>
{
  for $a in fn:doc("kameras.xml")//Angebot
    where $a/Modell/@Megapixel > 8
    order by $a/@Preis
    return
      <Kamera Aid="{ $a/@Aid }" Preis="{ $a/@Preis }">
        {
          fn:string($a/Modell/@Megapixel)
        }
      </Kamera>
}
</VielePixel>
```

```
<VielePixel>
    <Kamera Aid="C4711" Preis="398,50">10.1</Kamera>
    <Kamera Aid="P0965" Preis="529,00">10.2</Kamera>
    <Kamera Aid="C4712" Preis="599,50">12.2</Kamera>
</VielePixel>
```

Aufgabe 6:

Gegeben sei die folgende Datenbanktabelle mit Namen **ANGEBOTE**.

AID	MARKE	MODELL	MPIXEL	BEMERKUNG	PREIS
C4711	Canon	EOS 400D Kit mit EF S 18-55	10.1	meistverkaufte digitale Spiegelreflex	398,50
C4712	Canon	EOS 450D Kit mit EF S 18-55	12.2	mit Live-View	599,50
N2927	Nikon	D40 Kit mit AF-S DX Nikkor 18-55	6.1	Modell schwarz, handliche Reisekamera	378,00
N2423	Nikon	D40 Kit mit AF-S DX Nikkor 18-55	6.1	Body und Objektiv silber, Restexemplar neu	358,00
P0965	Pentax	K 200 D Kit mit DA 18-55 AL II	10.2	Bildstabil., staub- und wassergeschuetzt	529,00

Vervollständigen Sie die folgende SQL/XML-Abfage so, daß das Dokument aus Aufgabe 1 geliefert wird. Dabei wird allerdings das Wurzelelement **`Angebote`** nicht produziert.

```
SELECT
  XMLELEMENT__________(
    NAME "Angebot",
    XMLATTRIBUTES____________(
      AID AS "Aid",
      MARKE AS "Marke",
      PREIS AS "Preis"
    ),
    XMLELEMENT________(
      NAME "Modell",
      XMLATTRIBUTES___________(
        MPIXEL AS "Megapixel"
      ),
      MODELL
    ),
    XMLELEMENT____________________(
      NAME "Bemerkung",
      BEMERKUNG
    )
  )
FROM ANGEBOTE____________;
```

Aufgabe 7:

Die Firma Foto trübe-linse GmbH&Co KG hat zwar kein so tolles Angebot an Kameras, dafür aber ein animiertes Logo. Das SVG-Dokument verwendet ein Zeichen aus der Font-Familie Webdings, das eine stilisierte Kamera darstellt, das in unserem Fall in zwei Exemplaren entlang der Linie fliegt. Für die Lücken im SVG-Dokument geben wir rechts eine Auswahl an. Füllen Sie die Lücken mit den richtigen Nummern, wobei einige Begriffe garnicht, andere mehrfach passen können!

①	repeatCount
②	dur
③	Foto
④	symbol
⑤	path
⑥	font-family
⑦	fill
⑧	canvas
⑨	stroke

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<svg xmlns="http://www.w3.org/2000/svg">
  <__⑤__ d="M500,100 A100,50 0 1,1 500,500 100,50 0 1,1 500,100" __⑨__="red"
      stroke-width="5" __⑦__="none"/>
  <text x="0" y="15" _________⑥_____________="Webdings" font-size="100"
      text-anchor="middle">&#xB5;
    <animateMotion path="M500,100 A100,50 0 1,1 500,500 100,50 0 1,1 500,100"
        _②__="30s" rotate="auto" ________①_________="indefinite" />
  </text>
  <text x="0" y="15" __________⑥_____________="Webdings" font-size="100"
      text-anchor="middle">&#xB5;
    <animateMotion path="M500,500 A100,50 0 1,1 500,100 100,50 0 1,1 500,500"
        _②__="30s" rotate="auto" ________①_________="indefinite" />
  </text>
  <text x="500" y="200" font-family="Times" font-size="60"
      text-anchor="middle" font-style="italic">____③____</text>
  <text x="500" y="310" font-family="Times" font-size="100"
      text-anchor="middle" font-style="italic">t r ü b e - l i n s e</text>
  <text x="500" y="420" font-family="Times" font-size="60"
      text-anchor="middle" font-style="italic">GmbH &amp;amp; Co. KG</text>
</svg>
```

Klausur zur Vorlesung „Einführung in XML“

Nachname: Vorname:

Matr.Nr.: Studiengang:

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 90 Minuten.

Aufgabe	**Punkte max.**	**Punkte erreicht**
1	4	
2	1+1+2+1	
3	6	
4	4	
5	4	
6	4	
7	4	
Summe	**31**	

Aufgabe 1:

Betrachten Sie das folgende Dokument. Ist es ein wohlgeformtes XML-Dokument? Wenn nein, streichen Sie die Fehler an.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Meteorologie Werte="aktuell"/>
<Messwerte vom="2009-02-25" Ort="Kassel" Uhrzeit=1200>
  <Luftdruck Einheit="hPa">
    1024
  </Luftdruck>
  <Niederschlag Einheit='mm/h'>
    0.0
  </Niederschlag>
  <Globale Strahlung Einheit = "W/qm">
    424
  </Globale Strahlung>
</MESSWERTE>
```

Aufgabe 2:

Wenn man nicht weiß, worüber man reden soll, redet man über das Wetter. Hier sind die Wetteraussichten der nächsten drei Tage als XML-Dokument.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="wetter.xsl" ?>
<Wetter vom="2009-02-25" Ort="Kassel">
  <Heute Himmel="wolkig">
    <Temperatur tief="1" hoch="6" />
    <Wind Staerke="20" Einheit="km/h">wechselnd</Wind>
    <Pollenflug>keiner</Pollenflug>
  </Heute>
  <Vorhersage Wochentag="Do" Himmel="bedeckt">
    <Temperatur tief="-2" hoch="4" />
    <Wind Staerke="25" Einheit="km/h">aus West</Wind>
  </Vorhersage>
  <Vorhersage Wochentag="Fr" Himmel="bedeckt">
    <Temperatur tief="-3" hoch="2" />
    <Wind Staerke="15" Einheit="km/h">aus Nordwest</Wind>
  </Vorhersage>
  <Vorhersage Wochentag="Sa" Himmel="heiter">
    <Temperatur tief="-8" hoch="-2" />
    <Wind Staerke="5" Einheit="km/h">aus Nord</Wind>
  </Vorhersage>
</Wetter>
```

Eine DTD für das Wetter-Dokument sieht wie folgt aus.

```
<!ELEMENT Wetter (Heute,Vorhersage+)>
<!ATTLIST Wetter vom CDATA #REQUIRED
                  Ort CDATA #REQUIRED>
<!ELEMENT Heute (Temperatur,Wind,Pollenflug?)>
<!ATTLIST Heute Himmel ( heiter | bedeckt | wolkig | Regen | Schnee )
               #IMPLIED>
<!ELEMENT Vorhersage (Temperatur,Wind)>
<!ATTLIST Vorhersage Wochentag CDATA #REQUIRED
                     Himmel CDATA #IMPLIED>
<!ELEMENT Temperatur EMPTY>
<!ATTLIST Temperatur tief CDATA #IMPLIED
                     hoch CDATA #IMPLIED>
<!ELEMENT Wind (#PCDATA)>
<!ATTLIST Wind Staerke CDATA #IMPLIED
               Einheit CDATA #FIXED "km/h">
<!ELEMENT Pollenflug (#PCDATA)>
```

(a) Welches Element darf weggelassen werden?

(b) Welche **Attribute** sind Pflichtattribute?

(c) Für das Element **`Vorhersage`** soll sowohl die Reihenfolge der Unterelemente **`Temperatur`**, **`Wind`** als auch **`Wind`**, **`Temperatur`** möglich sein. Geben Sie die geänderte Definition für die DTD an.

(d) Für das Attribut **`Staerke`** im Element **`Wind`** würden wir gerne numerische Werte (positiver Integer) vorschreiben. Wie geht das in einer DTD?

Aufgabe 3:

Ein XML-Schema zum Wetter-Dokument, das auf der DTD von oben aufbaut, sieht wie folgt aus. Füllen Sie die Lücken!

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<xsd:element name="Wetter" type="____________________"/>

<xsd:complexType name="__________________">
  <xsd:sequence>
    <xsd:element name="Heute" type="HeuteT"/>
    <xsd:element name="Vorhersage" type ="VorhersageT"
        minOccurs="__________" maxOccurs="__________________"/>
  </xsd:sequence>
  <xsd:attribute name="vom" type="xsd:date" use="required"/>
  <xsd:attribute name="Ort" type="_____________" use="required"/>
</xsd:complexType>

<xsd:complexType name="HeuteT">
  <xsd:sequence>
    <xsd:element name="Temperatur" type="TemperaturT"/>
    <xsd:element name="Wind" type="WindT"/>
    <xsd:element name="Pollenflug" type="____________________"
        minOccurs="_____________"/>
  </xsd:sequence>
  <xsd:attribute name="Himmel" type="____________________"/>
</xsd:complexType>

<xsd:_____________________________ name="VorhersageT">
  <xsd:sequence>
    <xsd:element name="Temperatur" type="TemperaturT"/>
    <xsd:element name="Wind" type="WindT"/>
  </xsd:sequence>
  <xsd:attribute name="Wochentag" type="xsd:string"/>
  <xsd:attribute name="Himmel" type="____________________"/>
</xsd:_____________________________>

<xsd:_____________________ name="HimmelT">
  <xsd:restriction base="_____________________________">
    <xsd:enumeration value="heiter"/>
```

```
    <xsd:enumeration value="bedeckt"/>
    <xsd:enumeration value="wolkig"/>
    <xsd:enumeration value="Regen"/>
    <xsd:enumeration value="Schnee"/>
  </xsd:restriction>
</xsd:_____________________>

<xsd:complexType name="_____________________________">
  <xsd:attribute name="tief" type="xsd:decimal"/>
  <xsd:attribute name="hoch" type="xsd:decimal"/>
</xsd:complexType>

<xsd:complexType name="WindT">
  <xsd:simpleContent>
    <xsd:extension ____________________="xsd:string">
      <xsd:____________ name="Staerke" type="xsd:positiveInteger"/>
      <xsd:____________ name="Einheit" type="xsd:string"
            fixed="km/h"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

</xsd:schema>
```

Aufgabe 4:

Die untenstehende Abbildung zeigt die HTML-Ausgabe des Wetter-Dokuments aus Aufgabe 2. Die Ausgabe wurde mit dem Stylesheet unten erzeugt. Zu jedem Himmelattributwert *x* liegt ein Bild *x*`.gif` vor. Vervollständigen Sie das Stylesheet!


```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet version='1.0'
  xmlns:_____________="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
    <head><title>Wetteraussichten</title></head>
    <body>
      <h1>Aussichten vom
        <xsl:value-of select="_____________________________"/> für
        <xsl:value-of select="_____________________________"/>
      </h1>
```

```
      <xsl:apply-templates select="Wetter" />
    </body>
  </html>
</xsl:template>

<xsl:template _____________________="Wetter">
  <table cellspacing="0" cellpadding="10">
    <xsl:_____________________________ select="Vorhersage" />
  </table>
</xsl:template>

<xsl:template _____________________="Vorhersage">
  <tr style="background-color: blue">
    <td colspan="2" style="color: white">
        <xsl:value-of select="_____________________________"/>
    </td>
  </tr>
  <tr style="background-color: silver">
    <td><xsl:____________________ select="Temperatur/@tief"/> /
        <xsl:____________________ select="Temperatur/@hoch"/>
    </td>
    <td><img src="{concat(_____________________,'.gif')}"/></td>
  </tr>
</xsl:template>

</xsl:stylesheet>
```

Aufgabe 5:

Die folgende XQuery soll auf dem Wetter-Dokument **`wetter.xml`** aus Aufgabe 2 laufen. Welche Ausgabe wird geliefert?

```
<BesteTage>
{
  let $doc := fn:doc("wetter.xml")
  for $k in $doc//Vorhersage
  let $m := fn:max($doc//Vorhersage/Temperatur/@hoch)
  where $k/Temperatur/@hoch = $m
  return
    <Tag TempNiedrig="{$k/Temperatur/@tief}"
         TempHoch="{$k/Temperatur/@hoch}">
      {fn:string($k/@Wochentag)}
    </Tag>
}
</BesteTage>
```

Aufgabe 6:

Gegeben sei die folgende Datenbanktabelle mit Namen **STAEDTEWETTER**.

STADT	TEMPERATUR	WETTER
Amsterdam	0	bedeckt
Bangkok	37	wolkig
Berlin	-1	wolkig
Las Palmas	21	bedeckt
Lissabon	12	heiter
Miami	26	heiter
Rom	12	bedeckt
Wien	-2	Schnee
Zuerich	-4	Schnee

Vervollständigen Sie die folgende SQL/XML-Abfrage so, daß die Ausgabe ganz unten geliefert wird. Gesucht sind Städte, in denen es wärmer als 15 Grad **oder** das Wetter heiter ist (da wären wir nämlich gerne). Beachten Sie die Sortierung nach Städtenamen.

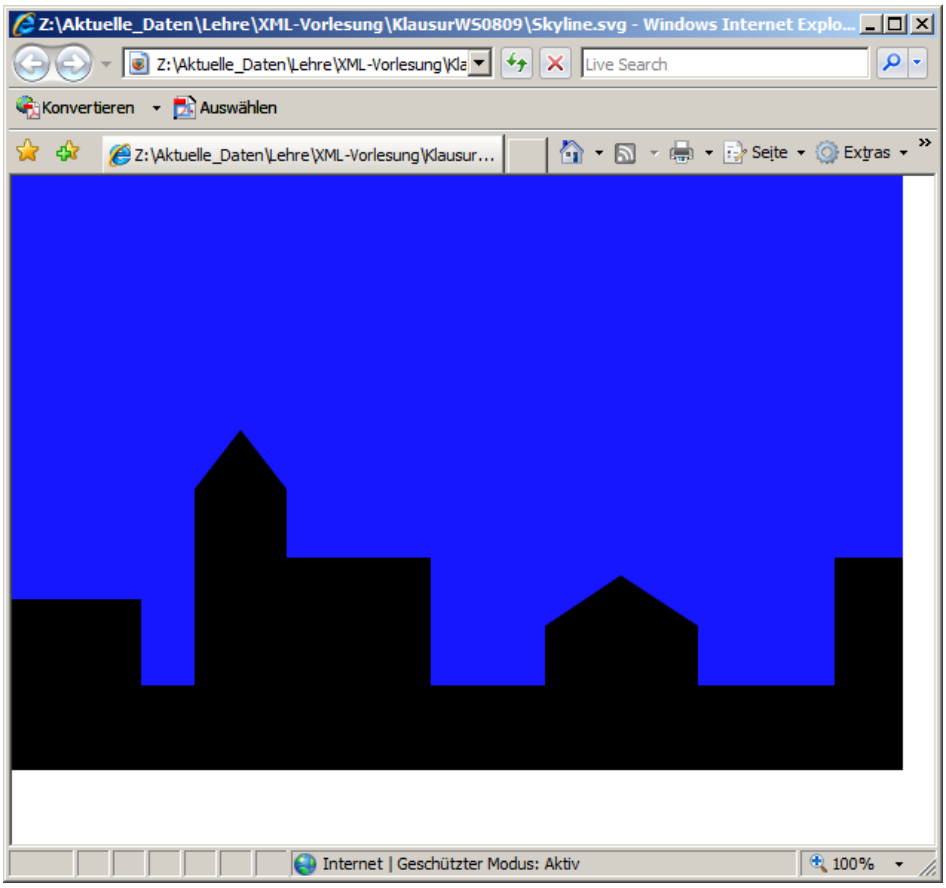
```
SELECT  XMLELEMENT (NAME "_____________________________",
        XMLATTRIBUTES (_____________________ AS "_____________",
                       _____________________ AS "_____________"),
        ____________________________________)
FROM STAEDTEWETTER
WHERE ____________________________________________________________________
ORDER BY _____________________________
```

Ausgabe

```
<Stadt Temp="37" Wetter="wolkig">Bangkok</Stadt>
<Stadt Temp="21" Wetter="bedeckt">Las Palmas</Stadt>
<Stadt Temp="12" Wetter="heiter">Lissabon</Stadt>
<Stadt Temp="26" Wetter="heiter">Miami</Stadt>
```

Aufgabe 7:

Es geht um eine animierte Graphik. Beschreiben Sie, was durch den Ablauf der Animation im SVG-Dokument unten angezeigt wird! Angabe auf der Rückseite.




```
<?xml version="1.0" encoding="ISO-8859-1"?>
<svg xmlns="http://www.w3.org/2000/svg">
  <rect x="0" y="0" width="600" height="400" fill="blue">
    <animateColor attributeName="fill" dur="10s" values="blue;white;blue"
          repeatCount="indefinite"/>
  </rect>

  <circle cx="-15" cy="-15" r="30" fill="yellow">
    <animateMotion path="M20,380 C 20,20 580,20 580,380" dur="10s"
          repeatCount="indefinite"/>
  </circle>

  <polygon points="0,400 0,285 87,285 87,343 123,343 123,211 154,171
        185,211 185,257 282,257 282,343 359,343 359,303 410,269
        462,303 462,343 554,343 554,257 600,257 600,400"/>

</svg>
```

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

Nachname: Vorname:

Matr.Nr.: Studiengang:

MUSTERLÖSUNG

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 90 Minuten.

Aufgabe	Punkte max.	Punkte erreicht
1	4	
2	1+1+2+1	
3	6	
4	4	
5	4	
6	4	
7	4	
Summe	**31**	

Aufgabe 1:

Betrachten Sie das folgende Dokument. Ist es ein wohlgeformtes XML-Dokument? Wenn nein, streichen Sie die Fehler an.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Meteorologie Werte="aktuell"/>
<Messwerte vom="2009-02-25" Ort="Kassel" Uhrzeit=1200>
  <Luftdruck Einheit="hPa">
    1024
  </Luftdruck>
  <Niederschlag Einheit='mm/h'>
    0.0
  </Niederschlag>
  <Globale Strahlung Einheit = "W/qm">
    424
  </Globale Strahlung>
</MESSWERTE>
```

zwei Wurzelelemente

Aufgabe 2:

Wenn man nicht weiß, worüber man reden soll, redet man über das Wetter. Hier sind die Wetteraussichten der nächsten drei Tage als XML-Dokument.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="wetter.xsl" ?>
<Wetter vom="2009-02-25" Ort="Kassel">
  <Heute Himmel="wolkig">
    <Temperatur tief="1" hoch="6" />
    <Wind Staerke="20" Einheit="km/h">wechselnd</Wind>
    <Pollenflug>keiner</Pollenflug>
  </Heute>
  <Vorhersage Wochentag="Do" Himmel="bedeckt">
    <Temperatur tief="-2" hoch="4" />
    <Wind Staerke="25" Einheit="km/h">aus West</Wind>
  </Vorhersage>
  <Vorhersage Wochentag="Fr" Himmel="bedeckt">
    <Temperatur tief="-3" hoch="2" />
    <Wind Staerke="15" Einheit="km/h">aus Nordwest</Wind>
  </Vorhersage>
  <Vorhersage Wochentag="Sa" Himmel="heiter">
    <Temperatur tief="-8" hoch="-2" />
    <Wind Staerke="5" Einheit="km/h">aus Nord</Wind>
  </Vorhersage>
</Wetter>
```

Eine DTD für das Wetter-Dokument sieht wie folgt aus.

```
<!ELEMENT Wetter (Heute,Vorhersage+)>
<!ATTLIST Wetter vom CDATA #REQUIRED
                  Ort CDATA #REQUIRED>
<!ELEMENT Heute (Temperatur,Wind,Pollenflug?)>
<!ATTLIST Heute Himmel ( heiter | bedeckt | wolkig | Regen | Schnee )
               #IMPLIED>
<!ELEMENT Vorhersage (Temperatur,Wind)>
<!ATTLIST Vorhersage Wochentag CDATA #REQUIRED
                     Himmel CDATA #IMPLIED>
<!ELEMENT Temperatur EMPTY>
<!ATTLIST Temperatur tief CDATA #IMPLIED
                     hoch CDATA #IMPLIED>
<!ELEMENT Wind (#PCDATA)>
<!ATTLIST Wind Staerke CDATA #IMPLIED
               Einheit CDATA #FIXED "km/h">
<!ELEMENT Pollenflug (#PCDATA)>
```

(a) Welches Element darf weggelassen werden?

Pollenflug

(b) Welche **Attribute** sind Pflichtattribute?

vom, Ort, Wochentag

(c) Für das Element **`Vorhersage`** soll sowohl die Reihenfolge der Unterelemente **`Temperatur`**, **`Wind`** als auch **`Wind`**, **`Temperatur`** möglich sein. Geben Sie die geänderte Definition für die DTD an.

<!ELEMENT Vorhersage ((Temperatur, Wind) | (Wind, Temperatur))>

(d) Für das Attribut **`Staerke`** im Element **`Wind`** würden wir gerne numerische Werte (positiver Integer) vorschreiben. Wie geht das in einer DTD?

Geht nicht!

Aufgabe 3:

Ein XML-Schema zum Wetter-Dokument, das auf der DTD von oben aufbaut, sieht wie folgt aus. Füllen Sie die Lücken!

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<xsd:element name="Wetter" type="__WetterT____"/>
                                                  oder ein anderer Bezeichner
<xsd:complexType name="___WetterT__">
  <xsd:sequence>
    <xsd:element name="Heute" type="HeuteT"/>
    <xsd:element name="Vorhersage" type ="VorhersageT"
        minOccurs="___1___" maxOccurs="___unbounded_____"/>
  </xsd:sequence>
  <xsd:attribute name="vom" type="xsd:date" use="required"/>
  <xsd:attribute name="Ort" type="_xsd:string__" use="required"/>
</xsd:complexType>

<xsd:complexType name="HeuteT">
  <xsd:sequence>
    <xsd:element name="Temperatur" type="TemperaturT"/>
    <xsd:element name="Wind" type="WindT"/>
    <xsd:element name="Pollenflug" type="__xsd:string__"
        minOccurs="____0_____"/>
  </xsd:sequence>
  <xsd:attribute name="Himmel" type="___HimmelT___"/>
</xsd:complexType>

<xsd:___complexType____ name="VorhersageT">
  <xsd:sequence>
    <xsd:element name="Temperatur" type="TemperaturT"/>
    <xsd:element name="Wind" type="WindT"/>
  </xsd:sequence>
  <xsd:attribute name="Wochentag" type="xsd:string"/>
  <xsd:attribute name="Himmel" type="____HimmelT___"/>
</xsd:____complexType____>
                                                  oder auch xsd:string

<xsd:___simpleType____ name="HimmelT">
  <xsd:restriction base="____xsd:string_____">
    <xsd:enumeration value="heiter"/>
```

```
    <xsd:enumeration value="bedeckt"/>
    <xsd:enumeration value="wolkig"/>
    <xsd:enumeration value="Regen"/>
    <xsd:enumeration value="Schnee"/>
  </xsd:restriction>
</xsd:____simpleType_____>

<xsd:complexType name="___TemperaturT___">
  <xsd:attribute name="tief" type="xsd:decimal"/>
  <xsd:attribute name="hoch" type="xsd:decimal"/>
</xsd:complexType>

<xsd:complexType name="WindT">
  <xsd:simpleContent>
    <xsd:extension ____base____="xsd:string">
      <xsd:attribute__ name="Staerke" type="xsd:positiveInteger"/>
      <xsd:attribute__ name="Einheit" type="xsd:string"
            fixed="km/h"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

</xsd:schema>
```

Aufgabe 4:

Die untenstehende Abbildung zeigt die HTML-Ausgabe des Wetter-Dokuments aus Aufgabe 2. Die Ausgabe wurde mit dem Stylesheet unten erzeugt. Zu jedem Himmelattributwert *x* liegt ein Bild *x*`.gif` vor. Vervollständigen Sie das Stylesheet!


```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet version='1.0'
  xmlns:xsl___="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
    <head><title>Wetteraussichten</title></head>
    <body>
      <h1>Aussichten vom
        <xsl:value-of select="__Wetter/@vom___"/> für
        <xsl:value-of select="__Wetter/@Ort___"/>
      </h1>
```

```
      <xsl:apply-templates select="Wetter" />
    </body>
  </html>
</xsl:template>

<xsl:template ____match____="Wetter">
  <table cellspacing="0" cellpadding="10">
    <xsl:apply-templates________ select="Vorhersage" />
  </table>
</xsl:template>

<xsl:template ___match____="Vorhersage">
  <tr style="background-color: blue">
    <td colspan="2" style="color: white">
        <xsl:value-of select="___@Wochentag____"/>
    </td>
  </tr>
  <tr style="background-color: silver">
    <td><xsl:value-of___ select="Temperatur/@tief"/> /
        <xsl:value-of__ select="Temperatur/@hoch"/>
    </td>
    <td><img src="{concat(_@Himmel_____,'.gif')}"/></td>
  </tr>
</xsl:template>

</xsl:stylesheet>
```

Aufgabe 5:

Die folgende XQuery soll auf dem Wetter-Dokument **`wetter.xml`** aus Aufgabe 2 laufen. Welche Ausgabe wird geliefert?

```
<BesteTage>
{
  let $doc := fn:doc("wetter.xml")
  for $k in $doc//Vorhersage
  let $m := fn:max($doc//Vorhersage/Temperatur/@hoch)
  where $k/Temperatur/@hoch = $m
  return
    <Tag TempNiedrig="{$k/Temperatur/@tief}"
         TempHoch="{$k/Temperatur/@hoch}">
      {fn:string($k/@Wochentag)}
    </Tag>
}
</BesteTage>
```

```
<BesteTage>
   <Tag TempNiedrig="-2" TempHoch="4">
       Do
   </Tag>
</BesteTage>
```

Aufgabe 6:

Gegeben sei die folgende Datenbanktabelle mit Namen **STAEDTEWETTER**.

STADT	TEMPERATUR	WETTER
Amsterdam	0	bedeckt
Bangkok	37	wolkig
Berlin	-1	wolkig
Las Palmas	21	bedeckt
Lissabon	12	heiter
Miami	26	heiter
Rom	12	bedeckt
Wien	-2	Schnee
Zuerich	-4	Schnee

Vervollständigen Sie die folgende SQL/XML-Abfrage so, daß die Ausgabe ganz unten geliefert wird. Gesucht sind Städte, in denen es wärmer als 15 Grad **oder** das Wetter heiter ist (da wären wir nämlich gerne). Beachten Sie die Sortierung nach Städtenamen.

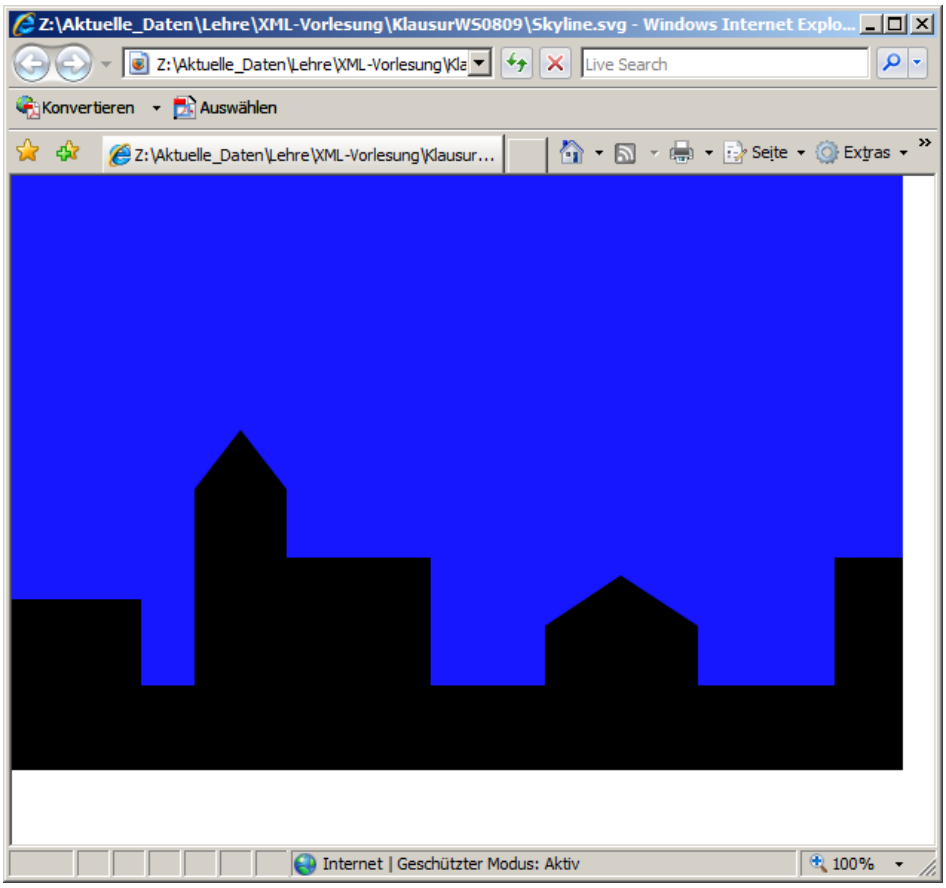
```
SELECT  XMLELEMENT (NAME "_____Stadt_____",
         XMLATTRIBUTES (_TEMPERATUR__ AS "__Temp__",
                        __WETTER_____ AS "__Wetter___"),
         ___STADT____)
FROM STAEDTEWETTER
WHERE ___TEMPERATUR > 15 OR WETTER = 'heiter'_____
ORDER BY __STADT_____
```

Ausgabe

```
<Stadt Temp="37" Wetter="wolkig">Bangkok</Stadt>
<Stadt Temp="21" Wetter="bedeckt">Las Palmas</Stadt>
<Stadt Temp="12" Wetter="heiter">Lissabon</Stadt>
<Stadt Temp="26" Wetter="heiter">Miami</Stadt>
```

Aufgabe 7:

Es geht um eine animierte Graphik. Beschreiben Sie, was durch den Ablauf der Animation im SVG-Dokument unten angezeigt wird! Angabe auf der Rückseite.



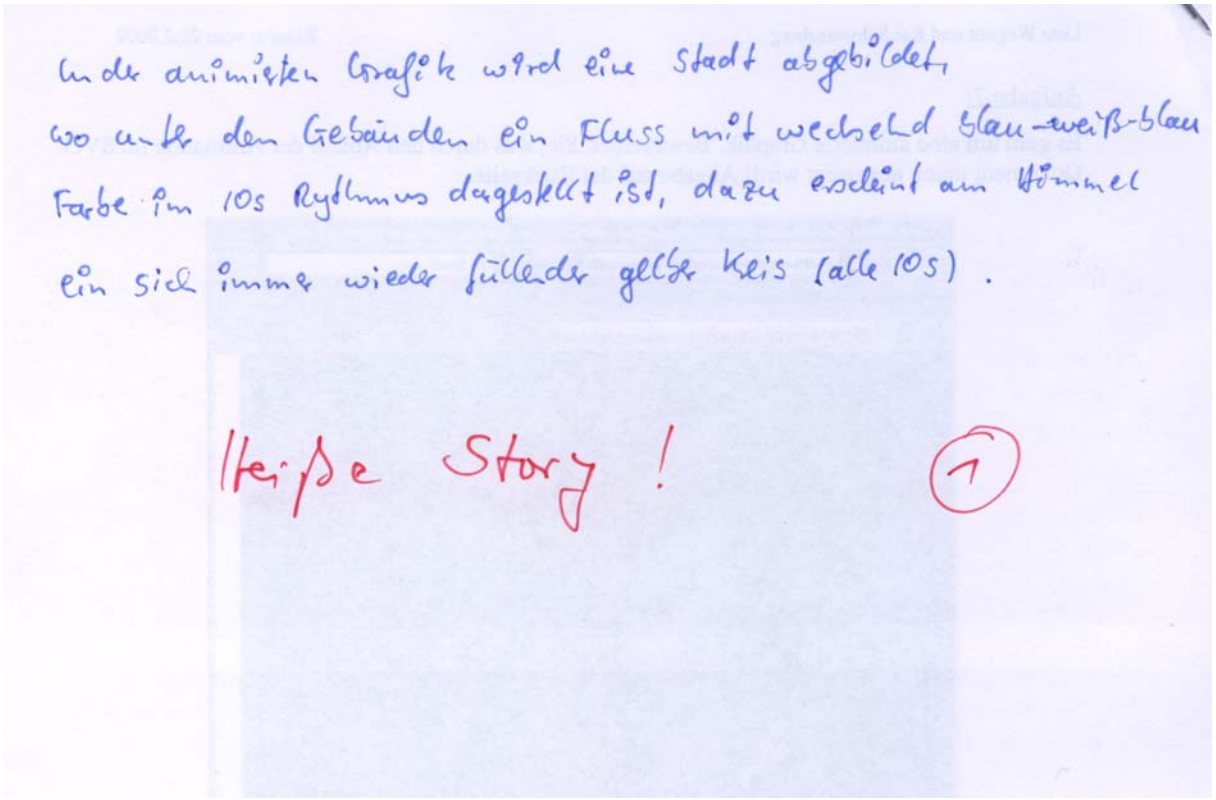

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<svg xmlns="http://www.w3.org/2000/svg">
  <rect x="0" y="0" width="600" height="400" fill="blue">
    <animateColor attributeName="fill" dur="10s" values="blue;white;blue"
          repeatCount="indefinite"/>
  </rect>

  <circle cx="-15" cy="-15" r="30" fill="yellow">
    <animateMotion path="M20,380 C 20,20 580,20 580,380" dur="10s"
          repeatCount="indefinite"/>
  </circle>

  <polygon points="0,400 0,285 87,285 87,343 123,343 123,211 154,171
        185,211 185,257 282,257 282,343 359,343 359,303 410,269
        462,303 462,343 554,343 554,257 600,257 600,400"/>

</svg>
```

ENDE DER KLAUSUR

Lösung siehe nächste Seite

Ein gelber Kreis (Sonne) bewegt sich bogenförmig von links nach rechts innerhalb von 10 Sekunden. Der Kreis erscheint und verschwindet hinter dem schwarzen Polygonzug (den Häusern). Im selben Intervall färbt sich der Hintergrund (Himmel) von Blau zu Weiß und wieder zu Blau. Die Animation wiederholt sich unendlich.

Fundstück



Klausur zur Vorlesung „Einführung in XML“

Nachname: Vorname:

Matr.Nr.: Studiengang:

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 120 Minuten.

Aufgabe	**Punkte max.**	**Punkte erreicht**
1	5 + 1	
2	2 + 2 + 2 + 2	
3	6 + 4	
4	6	
5	6	
6	6	
7	6	
Summe	**48**	

Aufgabe 1:

Wir verkaufen unser Oma ihr klein Häuschen! Betrachten Sie das Dokument *Angebote.xml* mit einer Liste von Immobilien (Häuser, Eigentumswohnungen, Bauplätze) zum Verkauf.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="AngebotsStyle.xsl" ?>
<!DOCTYPE Immobilien SYSTEM "Angebote.dtd">
<Immobilien OrtsId="34">
  <Haus Typ="EFH" Baujahr="1968" Preis="180000">
    <Lage>Kaufungen</Lage>
    <Grundstueck qm="850">mit Fernblick</Grundstueck>
    <Wohnen Zimmer="5" qm="140"/>
    <Bemerkung>Renovierungsstau, Bezug sofort</Bemerkung>
  </Haus>
  <Haus Typ="DHH" Baujahr="1992" Preis="153000">
    <Lage>Oberzwehren</Lage>
    <Grundstueck qm="550">linke Doppelhaushälfte</Grundstueck>
    <Wohnen Zimmer="4" qm="120"/>
    <Bemerkung>ruhige Sackgasse, huebscher Schnitt</Bemerkung>
  </Haus>
  <Haus Typ="MFH" Baujahr="1953" Preis="245000">
    <Lage>Bad Wilhelmshöhe</Lage>
    <Grundstueck qm="1030">gut eingewachsen</Grundstueck>
    <Wohnen Zimmer="5" qm="105">Wohnung 1</Wohnen>
    <Wohnen Zimmer="4" qm="90">Wohnung 2</Wohnen>
    <Bemerkung>noch vermietet</Bemerkung>
  </Haus>
  <Bauplatz Typ="BPL" Groesse="700" Preis="90000">
    <Lage>Grebenstein OT</Lage>
    <Bemerkung>erschlossen, bis 2FH</Bemerkung>
  </Bauplatz>
</Immobilien>
```

(a) Gemäß der DTD aus Aufgabe 2 kann die Liste auch Eigentumswohnungen enthalten. Geben Sie hier ein solches Element mit **allen** erlaubten Attributen und Unterelementen an! Dazu passende Werte erfinden Sie frei!

(b) Würde man im Element **`Haus`** die Reihenfolge der Attribute **`Typ`**, **`Baujahr`** und **`Preis`** vertauschen, dann müßte man das in allen Haus-Elementen machen?

Aufgabe 2:

Eine DTD für die Immobilien aus Aufgabe 1 sieht wie folgt aus.

```
<!-- DTD fuer die Immobilienliste -->
<!ELEMENT Immobilien (Haus | Eigentumswohnung | Bauplatz)*>
<!ATTLIST Immobilien OrtsId CDATA #REQUIRED>

<!ELEMENT Haus (Lage, Grundstueck, (Wohnen)+, Bemerkung)>
<!ATTLIST Haus Typ ( EFH | MFH | DHH | Bauernhof | ohne ) "ohne"
               Baujahr CDATA #REQUIRED
               Preis CDATA #IMPLIED>

<!ELEMENT Eigentumswohnung (Lage, Wohnen, Bemerkung)>
<!ATTLIST Eigentumswohnung Typ CDATA #FIXED "ETW"
                           Baujahr CDATA #REQUIRED
                           Preis CDATA #IMPLIED>

<!ELEMENT Bauplatz (Lage, Bemerkung)>
<!ATTLIST Bauplatz Typ CDATA #FIXED "BPL"
                   Groesse CDATA #REQUIRED
                   Preis CDATA #REQUIRED>

<!ELEMENT Lage (#PCDATA)>
<!ELEMENT Wohnen (#PCDATA)>
<!ATTLIST Wohnen Zimmer CDATA #REQUIRED
                 qm CDATA #REQUIRED>
<!ELEMENT Grundstueck (#PCDATA)>
<!ATTLIST Grundstueck qm CDATA #IMPLIED>
<!ELEMENT Bemerkung (#PCDATA)>
```

(a) Wie könnte man bestimmen, dass für das Attribut **qm** nur numerische Werte erlaubt sind?

(b) Welche **Attribute** sind im Element **Haus** optional?

(c) Welche Elemente können leer sein?

(e) Geben Sie in der DTD bei **Haus** ein Pflichtattribut **immoid** an, das vom Typ **ID** ist. Was wäre damit sichergestellt? Bitte Antwort auf der Rückseite!

Aufgabe 3:

Ein XML-Schema zum Angebote-Dokument sieht wie folgt aus.

(a) Füllen Sie die Lücken, wobei „Immobilien“ keine oder unbegrenzt viele Einträge enthalten, ein Haus mindestens eine und maximal 100 Wohnungen haben kann. Orientieren Sie sich bei den gewünschten Angaben an der DTD aus Aufgabe 2.

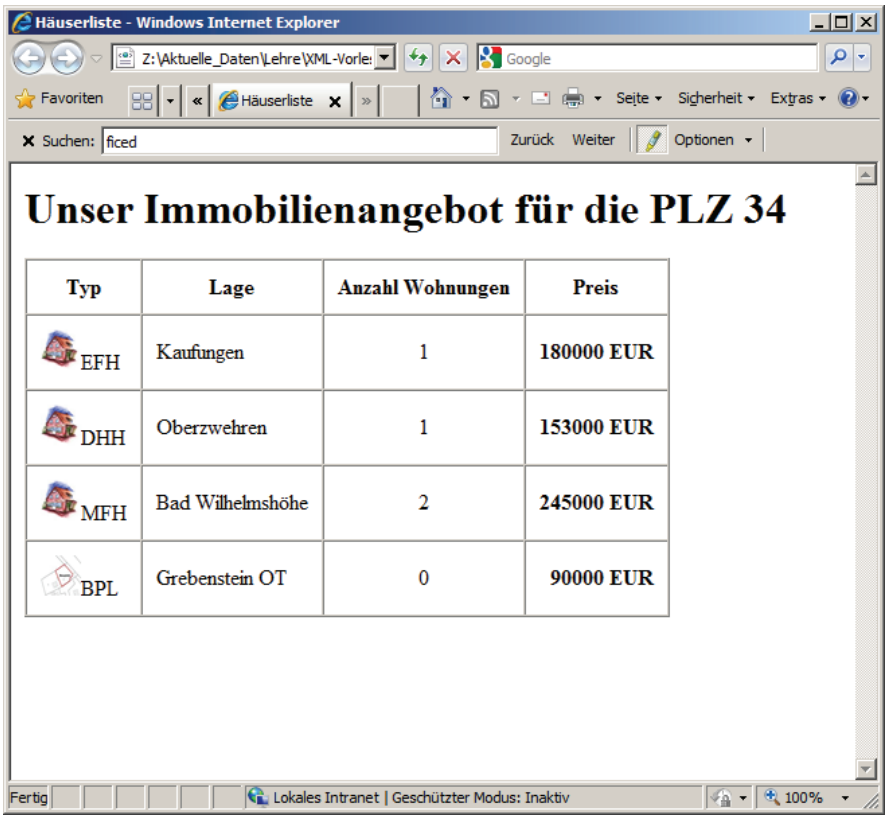
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<xsd:element name="Immobilien" type="ImmobilienT"/>

<xsd:complexType name="ImmobilienT">
  <xsd:choice minOccurs="_____" maxOccurs="____________________">
    <xsd:element name="Haus" type="HausT"/>
    <xsd:element name="Eigentumswohnung" type="_____________"/>
    <xsd:element name="Bauplatz" type="BauplatzT"/>
  </xsd:choice>
  <xsd:attribute name="OrtsId" type="xsd:integer" use="required"/>
</xsd:complexType>

 <xsd:complexType name="HausT">
  <xsd:sequence>
    <xsd:element name="Lage" type="xsd:string"/>
    <xsd:element name="Grundstueck" type="GrundstueckT"/>
    <xsd:element name="Wohnen" type="WohnenT"
                 minOccurs="______" maxOccurs="_______"/>
    <xsd:element name="Bemerkung" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="Typ" default="_____________">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="EFH"/>
        <xsd:enumeration value="MFH"/>
        <xsd:enumeration value="DHH"/>
        <xsd:enumeration value="Bauernhof"/>
        <xsd:enumeration value="ohne"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="Baujahr" type="BaujahrT" use="____________"/>
  <xsd:attribute name="Preis" type="xsd:positiveInteger"/>
</xsd:complexType>

<xsd:complexType name="ETWT">
  <xsd:sequence>
    <xsd:element name="_________________" type="xsd:string"/>
    <xsd:element name="Wohnen" type="WohnenT"/>
    <xsd:element name="_________________" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="Typ" type="xsd:string" _____________="ETW"/>
  <xsd:attribute name="Baujahr" type="BaujahrT" use="required"/>
  <xsd:attribute name="Preis" type="xsd:positiveInteger"/>
</xsd:complexType>
```

```
<xsd:complexType name="BauplatzT">
  <xsd:sequence>
    <xsd:element name="Lage" type="xsd:string"/>
    <xsd:element name="Bemerkung" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="Typ" type="xsd:string" _____="BPL"/>
  <xsd:attribute name="Groesse" type="xsd:decimal" use="required"/>
  <xsd:attribute name="Preis" type="xsd:positiveInteger"
        use="required"/>
</xsd:complexType>

<xsd:complexType name="WohnenT">
  <xsd:_____>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="Zimmer" type="xsd:decimal"
            use="required"/>
      <xsd:attribute name="qm" type="xsd:decimal" use="required"/>
    </xsd:extension>
  </xsd:_____>
</xsd:complexType>

<xsd:complexType name="GrundstueckT">
  <xsd:_____>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="qm" type="xsd:decimal"/>
    </xsd:extension>
  </xsd:_____>
</xsd:complexType>

<xsd:simpleType name="BaujahrT">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[1-9][_____]{3}"/>
  </xsd:restriction>
</xsd:simpleType>

</xsd:schema>
```

(b) Nehmen wir an, in Häusern gebe es neben dem Element **`Wohnen`** alternativ auch das Element **`Praxis`**, wenn das Haus neben „normalen“ Wohnungen auch Praxen für Ärzte, Anwälte usw. aufweist. In der DTD stünde dann
`<!ELEMENT Haus (Lage, Grundstueck, (Wohnen | Praxis)+, Bemerkung)>`
Wie wäre **`HausT`** oben zu verändern, damit das XML-Schema dies ausdrückt? Geben Sie nur die neuen Teile an. Beachten Sie, dass bis zu 100 Wohnungen und Praxen insgesamt erlaubt sind.

Aufgabe 4:

Die untenstehende Abbildung zeigt die HTML-Ausgabe des XML-Dokuments zu den Angeboten aus Aufgabe 1. Die Ausgabe wurde mit dem Stylesheet unten erzeugt. Vervollständigen Sie das Stylesheet!




```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet version='1.0'
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
    <head><title>Häuserliste</title></head>
    <body>
      <h1>Unser Immobilienangebot für die PLZ
          <xsl:value-of select="_____"/></h1>
      <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>
```

```
<xsl:template match="Immobilien">
  <table border="1" cellspacing="0" cellpadding="10">
    <tr>
    <th>Typ</th><th>Lage</th><th>Anzahl Wohnungen</th><th>Preis</th>
    </tr>

      ____________________________________________________________

  </table>
</xsl:template>

<xsl:template match="Haus| ___________________ |Eigentumswohnung">
  <tr>
    <td>
      <img width="30" height="30">
      <xsl:attribute name="src">
        <xsl:if test="name()='Haus'">Bild1.jpg</xsl:if>
        <xsl:if test="name()='_________________'">Bild2.jpg</xsl:if>
        <xsl:if test="name()='Eigentumswohnung'">Bild3.jpg</xsl:if>
      </xsl:attribute>
      </img>
      <xsl:value-of select="_____________________"/>
    </td>
    <td><xsl:value-of select="_________________"/></td>
    <td align="center"><xsl:value-of select="count(____________)"/>
    </td>
    <td align="right"><b><xsl:value-of select="_________"/> EUR</b>
    </td>
  </tr>
</xsl:template>

</xsl:stylesheet>
```

Aufgabe 5:

Die folgende XQuery soll auf dem Dokument **`Angebote.xml`** aus Aufgabe 1 laufen. Ergänzen Sie die Lücken!

Hinweis: **fn:round** rundet eine Gleitkommazahl auf eine ganze Zahl. Die Multiplikation und folgende Division mit 100 wird benötigt, um genau zwei Nachkommastellen (Euro und Cent) für den Preis pro Quadratmeter Wohnfläche (QP) im Ergebnis zu haben. GF ist die Gesamtwohnfäche.

```
<Angebote>
{
  let $doc := fn:doc("Angebote.xml")
  for $haus in $doc/Immobilien/Haus
    return
    <Haus>
      <GF>
        {
          fn:sum(_____________________________________________)
        }
      </GF>
      <QP>
        {
          fn:round((___________________ div

                  fn:sum(_____________________________________))*100) div 100
        }
      </QP>
  </Haus>
}
</Angebote>
```

Ergebnis:

```
<Angebote>
    <Haus>
        <GF>140</GF>
        <QP>1285.71</QP>
    </Haus>
    <Haus>
        <GF>120</GF>
        <QP>1275</QP>
    </Haus>
    <Haus>
        <GF>195</GF>
        <QP>1256.41</QP>
    </Haus>
</Angebote>
```

Aufgabe 6:

Gegeben sei die folgende Datenbanktabelle mit Namen **RENDITEOBJEKTE**.

LAGE	TYP	WOHNUNGEN	PREIS
Innenstadt	MFH	4	400000
Südstadt	MFH	6	480000
Lohfelden	DHH	2	280000
Brasselsberg	EFH	1	450000

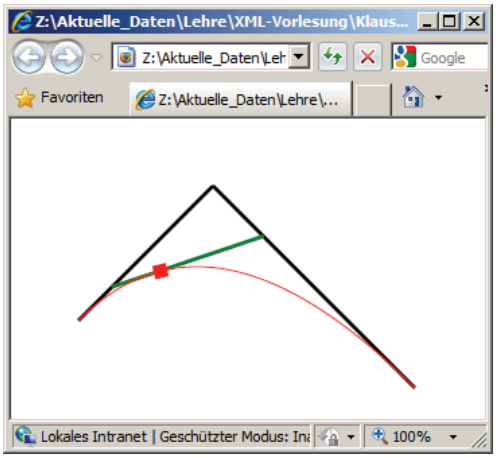
Geben Sie das Ergebnis der untenstehenden SQL/XML-Abfrage an!

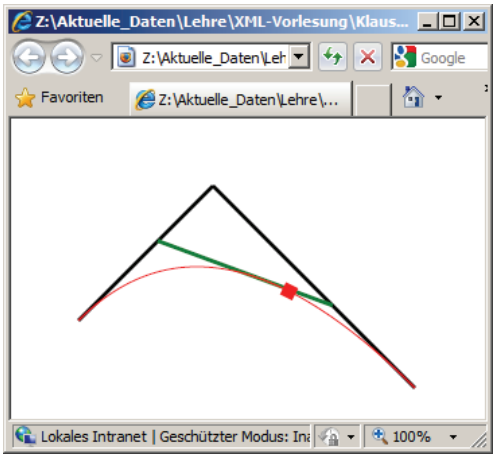
```
SELECT XMLELEMENT (
  NAME "Immobilie",
  XMLELEMENT (NAME "Lage", LAGE),
  XMLELEMENT (NAME "PreisProWohnung",
    XMLATTRIBUTES (Typ AS "Haustyp"),
    PREIS / WOHNUNGEN))
FROM RENDITEOBJEKTE
WHERE WOHNUNGEN > 3;
```

Ergebnis:

Aufgabe 7:

Keine XML-Klausur ohne eine SVG-Aufgabe. Hier geht es um eine Animation für eine quadratische Bézier-Kurve. Start- und Endpunkt der **grünen** Tangente laufen entlang der beiden schwarzen Geraden. Das rote Quadrat dreht sich automatisch auf dem Kurvenpfad. Füllen Sie die Lücken im SVG-Dokument aus!






```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg">
  <line style="stroke-width:3;fill:none; stroke:black"
      x1="50" y1="150" x2="150" y2="50"/>
  <line style="stroke-width:3;fill:none; stroke:black"
      x1="150" x2="300" y1="50" y2="200"/>
  <line style="stroke-width:3;fill:none; stroke:_____">
    <animate attributeName="x1" from="____" to="____" dur="5s"
        repeatCount="indefinite"/>
    <animate attributeName="y1" from="150" to="50" dur="5s"
        repeatCount="indefinite"/>
    <animate attributeName="x2" from="150" to="300" dur="____"
        repeatCount="indefinite"/>
    <animate attributeName="y2" from="____" to="____" dur="____"
        repeatCount="indefinite"/>
  </line>
  <path d="M50, 150 Q150, 50 300, 200" stroke="red" fill="none"/>
  <rect x="-5" y="-5" width="10" height="10" fill="red">
    <animateMotion __________="M50, 150 Q150, 50 300, 200"
        rotate="________" dur="5s" repeatCount="indefinite"/>
  </rect>
</svg>
```

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

Nachname: Vorname:

Matr.Nr.: Studiengang:

MUSTERLÖSUNG

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 120 Minuten.

Aufgabe	Punkte max.	Punkte erreicht
1	5 + 1	
2	2 + 2 + 2 + 2	
3	6 + 4	
4	6	
5	6	
6	6	
7	6	
Summe	**48**	

Aufgabe 1:

Wir verkaufen unser Oma ihr klein Häuschen! Betrachten Sie das Dokument *Angebote.xml* mit einer Liste von Immobilien (Häuser, Eigentumswohnungen, Bauplätze) zum Verkauf.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="AngebotsStyle.xsl" ?>
<!DOCTYPE Immobilien SYSTEM "Angebote.dtd">
<Immobilien OrtsId="34">
  <Haus Typ="EFH" Baujahr="1968" Preis="180000">
    <Lage>Kaufungen</Lage>
    <Grundstueck qm="850">mit Fernblick</Grundstueck>
    <Wohnen Zimmer="5" qm="140"/>
    <Bemerkung>Renovierungsstau, Bezug sofort</Bemerkung>
  </Haus>
  <Haus Typ="DHH" Baujahr="1992" Preis="153000">
    <Lage>Oberzwehren</Lage>
    <Grundstueck qm="550">linke Doppelhaushälfte</Grundstueck>
    <Wohnen Zimmer="4" qm="120"/>
    <Bemerkung>ruhige Sackgasse, huebscher Schnitt</Bemerkung>
  </Haus>
  <Haus Typ="MFH" Baujahr="1953" Preis="245000">
    <Lage>Bad Wilhelmshöhe</Lage>
    <Grundstueck qm="1030">gut eingewachsen</Grundstueck>
    <Wohnen Zimmer="5" qm="105">Wohnung 1</Wohnen>
    <Wohnen Zimmer="4" qm="90">Wohnung 2</Wohnen>
    <Bemerkung>noch vermietet</Bemerkung>
  </Haus>
  <Bauplatz Typ="BPL" Groesse="700" Preis="90000">
    <Lage>Grebenstein OT</Lage>
    <Bemerkung>erschlossen, bis 2FH</Bemerkung>
  </Bauplatz>
</Immobilien>
```

(a) Gemäß der DTD aus Aufgabe 2 kann die Liste auch Eigentumswohnungen enthalten. Geben Sie hier ein solches Element mit **allen** erlaubten Attributen und Unterelementen an! Dazu passende Werte erfinden Sie frei!

```
<Eigentumswohnung Typ="ETW" Baujahr="2008" Preis="100000">
  <Lage>Wehlheiden</Lage>
  <Wohnen Zimmer="3" qm="80"/>
  <Bemerkung/>
</Eigentumswohnung>
```

andere Werte und nichtleere Elemente möglich!

(b) Würde man im Element **`Haus`** die Reihenfolge der Attribute **`Typ`**, **`Baujahr`** und **`Preis`** vertauschen, dann müßte man das in allen Haus-Elementen machen?

Nein, muß man nicht.

Aufgabe 2:

Eine DTD für die Immobilien aus Aufgabe 1 sieht wie folgt aus.

```
<!-- DTD fuer die Immobilienliste -->
<!ELEMENT Immobilien (Haus | Eigentumswohnung | Bauplatz)*>
<!ATTLIST Immobilien OrtsId CDATA #REQUIRED>

<!ELEMENT Haus (Lage, Grundstueck, (Wohnen)+, Bemerkung)>
<!ATTLIST Haus Typ ( EFH | MFH | DHH | Bauernhof | ohne ) "ohne"
               Baujahr CDATA #REQUIRED
               Preis CDATA #IMPLIED
                                        <-------- (*) immoid ID #REQUIRED>
<!ELEMENT Eigentumswohnung (Lage, Wohnen, Bemerkung)>
<!ATTLIST Eigentumswohnung Typ CDATA #FIXED "ETW"
                           Baujahr CDATA #REQUIRED
                           Preis CDATA #IMPLIED>

<!ELEMENT Bauplatz (Lage, Bemerkung)>
<!ATTLIST Bauplatz Typ CDATA #FIXED "BPL"
                   Groesse CDATA #REQUIRED
                   Preis CDATA #REQUIRED>

<!ELEMENT Lage (#PCDATA)>
<!ELEMENT Wohnen (#PCDATA)>
<!ATTLIST Wohnen Zimmer CDATA #REQUIRED
                 qm CDATA #REQUIRED>
<!ELEMENT Grundstueck (#PCDATA)>
<!ATTLIST Grundstueck qm CDATA #IMPLIED>
<!ELEMENT Bemerkung (#PCDATA)>
```

(a) Wie könnte man bestimmen, dass für das Attribut **qm** nur numerische Werte erlaubt sind?

geht nicht mit DTD

(b) Welche **Attribute** sind im Element **Haus** optional?

Typ und Preis

(c) Welche Elemente können leer sein?

Immobilien, Lage, Wohnen, Grundstueck, Bemerkung

(e) Geben Sie in der DTD bei **Haus** ein Pflichtattribut **immoid** an, das vom Typ **ID** ist. Was wäre damit sichergestellt? Bitte Antwort auf der Rückseite!

siehe (*) Jedes Haus hätte ein Attribut immoid und für keine zwei Häuser wäre der selbe immoid-Wert zugelassen.

Aufgabe 3:

Ein XML-Schema zum Angebote-Dokument sieht wie folgt aus.

(a) Füllen Sie die Lücken, wobei „Immobilien“ keine oder unbegrenzt viele Einträge enthalten, ein Haus mindestens eine und maximal 100 Wohnungen haben kann. Orientieren Sie sich bei den gewünschten Angaben an der DTD aus Aufgabe 2.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<xsd:element name="Immobilien" type="ImmobilienT"/>

<xsd:complexType name="ImmobilienT">
  <xsd:choice minOccurs="__0___" maxOccurs="_unbounded_">
    <xsd:element name="Haus" type="HausT"/>
    <xsd:element name="Eigentumswohnung" type="_ETWT_"/>
    <xsd:element name="Bauplatz" type="BauplatzT"/>
  </xsd:choice>
  <xsd:attribute name="OrtsId" type="xsd:integer" use="required"/>
</xsd:complexType>

 <xsd:complexType name="HausT">
  <xsd:sequence>
    <xsd:element name="Lage" type="xsd:string"/>
    <xsd:element name="Grundstueck" type="GrundstueckT"/>
    <xsd:element name="Wohnen" type="WohnenT"
                 minOccurs="__1__" maxOccurs="__100__"/>      (*)
    <xsd:element name="Bemerkung" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="Typ" default="__ohne__">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="EFH"/>
        <xsd:enumeration value="MFH"/>
        <xsd:enumeration value="DHH"/>
        <xsd:enumeration value="Bauernhof"/>
        <xsd:enumeration value="ohne"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="Baujahr" type="BaujahrT" use="__required__"/>
  <xsd:attribute name="Preis" type="xsd:positiveInteger"/>
</xsd:complexType>

<xsd:complexType name="ETWT">
  <xsd:sequence>
    <xsd:element name="_Lage_" type="xsd:string"/>
    <xsd:element name="Wohnen" type="WohnenT"/>
    <xsd:element name="_Bemerkung_" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="Typ" type="xsd:string" fixed="ETW"/>
  <xsd:attribute name="Baujahr" type="BaujahrT" use="required"/>
  <xsd:attribute name="Preis" type="xsd:positiveInteger"/>
</xsd:complexType>
```

```
<xsd:complexType name="BauplatzT">
  <xsd:sequence>
    <xsd:element name="Lage" type="xsd:string"/>
    <xsd:element name="Bemerkung" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="Typ" type="xsd:string" fixed="BPL"/>
  <xsd:attribute name="Groesse" type="xsd:decimal" use="required"/>
  <xsd:attribute name="Preis" type="xsd:positiveInteger"
        use="required"/>
</xsd:complexType>

<xsd:complexType name="WohnenT">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="Zimmer" type="xsd:decimal"
            use="required"/>
      <xsd:attribute name="qm" type="xsd:decimal" use="required"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="GrundstueckT">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="qm" type="xsd:decimal"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:simpleType name="BaujahrT">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[1-9][__0-9__]{3}"/>
  </xsd:restriction>
</xsd:simpleType>

</xsd:schema>
```

(b) Nehmen wir an, in Häusern gebe es neben dem Element **Wohnen** alternativ auch das Element **Praxis**, wenn das Haus neben „normalen“ Wohnungen auch Praxen für Ärzte, Anwälte usw. aufweist. In der DTD stünde dann
<!ELEMENT Haus (Lage, Grundstueck, (Wohnen | Praxis)+, Bemerkung)>
Wie wäre **HausT** oben zu verändern, damit das XML-Schema dies ausdrückt? Geben Sie nur die neuen Teile an. Beachten Sie, dass bis zu 100 Wohnungen und Praxen insgesamt erlaubt sind.

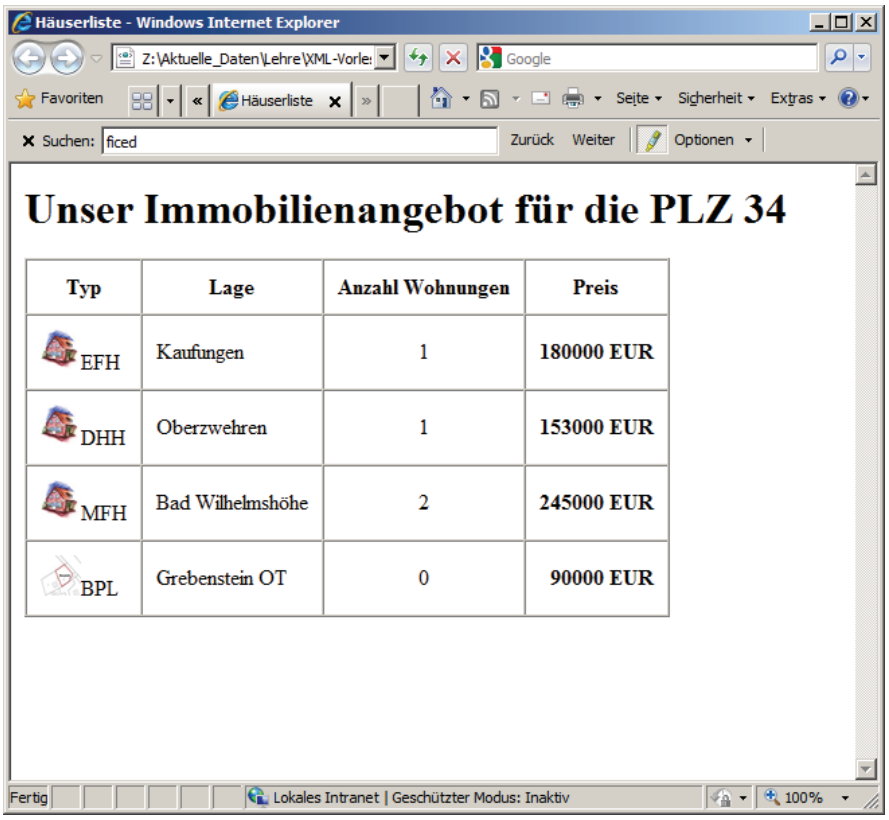
ersetze (*) durch

```
<xsd:choice minOccurs="1" maxOccurs="100">
  <xsd:element name="Wohnen" type="WohnenT"/>
  <xsd:element name="Praxis" type="..."/>
</xsd:choice>
```

als Typ für Praxis auch PraxisT oder WohnenT

Aufgabe 4:

Die untenstehende Abbildung zeigt die HTML-Ausgabe des XML-Dokuments zu den Angeboten aus Aufgabe 1. Die Ausgabe wurde mit dem Stylesheet unten erzeugt. Vervollständigen Sie das Stylesheet!




```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet version='1.0'
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
    <head><title>Häuserliste</title></head>
    <body>
      <h1>Unser Immobilienangebot für die PLZ
          <xsl:value-of select="Immobilien/@OrtsId"/></h1>
      <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>
```

```
<xsl:template match="Immobilien">
  <table border="1" cellspacing="0" cellpadding="10">
    <tr>
    <th>Typ</th><th>Lage</th><th>Anzahl Wohnungen</th><th>Preis</th>
    </tr>
      <xsl:apply-templates/>
  </table>
</xsl:template>

<xsl:template match="Haus| Bauplatz |Eigentumswohnung">
  <tr>
    <td>
      <img width="30" height="30">
      <xsl:attribute name="src">
        <xsl:if test="name()='Haus'">Bild1.jpg</xsl:if>
        <xsl:if test="name()='Bauplatz'">Bild2.jpg</xsl:if>
        <xsl:if test="name()='Eigentumswohnung'">Bild3.jpg</xsl:if>
      </xsl:attribute>
      </img>
      <xsl:value-of select="@Typ"/>
    </td>
    <td><xsl:value-of select="Lage"/></td>
    <td align="center"><xsl:value-of select="count(Wohnen)"/>
    </td>
    <td align="right"><b><xsl:value-of select="@Preis"/> EUR</b>
    </td>
  </tr>
</xsl:template>

</xsl:stylesheet>
```

Aufgabe 5:

Die folgende XQuery soll auf dem Dokument **`Angebote.xml`** aus Aufgabe 1 laufen. Ergänzen Sie die Lücken!

Hinweis: **fn:round** rundet eine Gleitkommazahl auf eine ganze Zahl. Die Multiplikation und folgende Division mit 100 wird benötigt, um genau zwei Nachkommastellen (Euro und Cent) für den Preis pro Quadratmeter Wohnfläche (QP) im Ergebnis zu haben. GF ist die Gesamtwohnfäche.

```
<Angebote>
{
  let $doc := fn:doc("Angebote.xml")
  for $haus in $doc/Immobilien/Haus
    return
    <Haus>
      <GF>
        {
          fn:sum($haus/Wohnen/@qm)
        }
      </GF>
      <QP>
        {
          fn:round(($haus/@Preis div
                 fn:sum($haus/Wohnen/@qm))*100) div 100
        }
      </QP>
  </Haus>
}
</Angebote>
```

Ergebnis:

```
<Angebote>
    <Haus>
        <GF>140</GF>
        <QP>1285.71</QP>
    </Haus>
    <Haus>
        <GF>120</GF>
        <QP>1275</QP>
    </Haus>
    <Haus>
        <GF>195</GF>
        <QP>1256.41</QP>
    </Haus>
</Angebote>
```

Aufgabe 6:

Gegeben sei die folgende Datenbanktabelle mit Namen **RENDITEOBJEKTE**.

LAGE	TYP	WOHNUNGEN	PREIS
Innenstadt	MFH	4	400000
Südstadt	MFH	6	480000
Lohfelden	DHH	2	280000
Brasselsberg	EFH	1	450000

Geben Sie das Ergebnis der untenstehenden SQL/XML-Abfrage an!

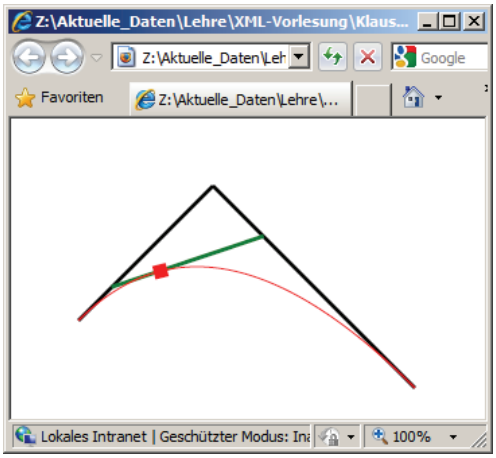
```
SELECT XMLELEMENT (
  NAME "Immobilie",
  XMLELEMENT (NAME "Lage", LAGE),
  XMLELEMENT (NAME "PreisProWohnung",
    XMLATTRIBUTES (Typ AS "Haustyp"),
    PREIS / WOHNUNGEN))
FROM RENDITEOBJEKTE
WHERE WOHNUNGEN > 3;
```

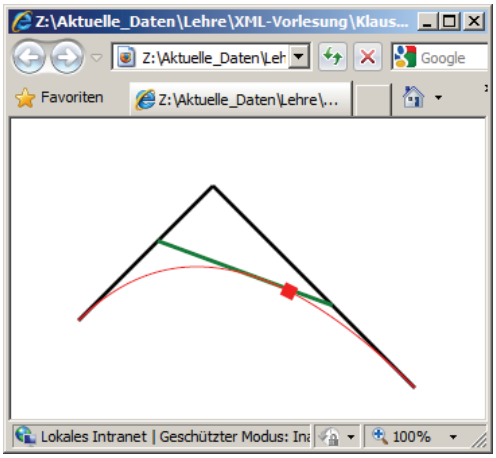
Ergebnis:

```
<Immobilie>
  <Lage>Innenstadt</Lage>
  <PreisProWohnung Haustyp="MFH">100000</PreisProWohnung>
</Immobilie>

<Immobilie>
  <Lage>Südstadt</Lage>
  <PreisProWohnung Haustyp="MFH">80000</PreisProWohnung>
</Immobilie>
```

Aufgabe 7:

Keine XML-Klausur ohne eine SVG-Aufgabe. Hier geht es um eine Animation für eine quadratische Bézier-Kurve. Start- und Endpunkt der **grünen** Tangente laufen entlang der beiden schwarzen Geraden. Das rote Quadrat dreht sich automatisch auf dem Kurvenpfad. Füllen Sie die Lücken im SVG-Dokument aus!






```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg">
  <line style="stroke-width:3;fill:none; stroke:black"
      x1="50" y1="150" x2="150" y2="50"/>
  <line style="stroke-width:3;fill:none; stroke:black"
      x1="150" x2="300" y1="50" y2="200"/>
  <line style="stroke-width:3;fill:none; stroke:green">
    <animate attributeName="x1" from="50" to="150" dur="5s"
        repeatCount="indefinite"/>
    <animate attributeName="y1" from="150" to="50" dur="5s"
        repeatCount="indefinite"/>
    <animate attributeName="x2" from="150" to="300" dur="5s"
        repeatCount="indefinite"/>
    <animate attributeName="y2" from="50" to="200" dur="5s"
        repeatCount="indefinite"/>
  </line>
  <path d="M50, 150 Q150, 50 300, 200" stroke="red" fill="none"/>
  <rect x="-5" y="-5" width="10" height="10" fill="red">
    <animateMotion path="M50, 150 Q150, 50 300, 200"
        rotate="auto" dur="5s" repeatCount="indefinite"/>
  </rect>
</svg>
```

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

Nachname: Vorname:

Matr.Nr.: Studiengang:

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 120 Minuten.

Aufgabe	**Punkte max.**	**Punkte erreicht**
1	6	
2	2 + 2 + 2 + 2	
3	4 + 4	
4	8	
5	6	
6	6	
7	6	
Summe	**48**	

Klausur **A**

Aufgabe 1:

Derzeit werden den Finanzbehörden von verschiedenen Seiten Datenträger mit gestohlenen Bankdaten angeboten. Jede Datensammlung hat ihr eigenes Format. Wir schlagen vor, mit einem einheitlichen Dokument *Hehlau.xml*, das auf jeden Datenträger gehört, Ordnung in dieses Chaos zu bringen. Eine DTD dafür finden Sie in Aufgabe 2.

Geben Sie hier ein zur DTD passendes XML-Dokument für die folgende Datensammlung an.

Das Datenangebot ist vom 25.02.2010 und kommt von einem männlichen schweizer Anbieter (Nationalitätsangabe CH) mit einer Preisvorstellung von 250000 €. Der Kontakt erfolgt über `Speedy4711@taxml.ky`. Als Datenvolumen will er 5000 Kunden liefern, der SWIFT-Bankidentifikationskode lautet `UBSWCHZH80A`, die Anzahl der Datensätze seien 50000 Stammdaten und 200000 Umsatzdaten aus dem Zeitraum 01/2008 bis 12/2009.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Datenangebot SYSTEM "Hehlau.dtd">
<Datenangebot Datum="__________" DatumsFormat="________________">




</Datenangebot>
```

Aufgabe 2:

Eine DTD für das Datenangebot aus Aufgabe 1 sieht wie folgt aus.

```
<!-- DTD fuer die DatenCD -->
<!ELEMENT Datenangebot (Anbieter, Volumen, Preis)>
<!ATTLIST Datenangebot Datum CDATA #REQUIRED
                       DatumsFormat CDATA #FIXED "DD.MM.YYYY">

<!ELEMENT Anbieter (Kontakt, Bemerkung*)>
<!ATTLIST Anbieter Geschlecht ( M | F | kA ) "kA"
                   Nationalitaet CDATA #IMPLIED>

<!ELEMENT Kontakt (#PCDATA)>

<!ELEMENT Bemerkung (#PCDATA)>

<!ELEMENT Volumen (AnzahlKunden, AnzahlSaetze+, Zeitraum?)>

<!ELEMENT AnzahlKunden (#PCDATA)>
<!ATTLIST AnzahlKunden SWIFT-BIC CDATA #REQUIRED>

<!ELEMENT AnzahlSaetze (#PCDATA)>
<!ATTLIST AnzahlSaetze Art ( Stamm | Umsatz | Schliessfach | mixed )
            "mixed">

<!ELEMENT Zeitraum EMPTY>
<!ATTLIST Zeitraum von CDATA #REQUIRED
                   bis CDATA #REQUIRED>

<!ELEMENT Preis (#PCDATA)>
<!ATTLIST Preis Waehrung ( EUR | USD | SFR ) "USD">
```

(a) Was bewirkt das Fragezeichen bei der Angabe des Elements **`Zeitraum?`** in **`Volumen`**?

(b) Was bewirkt die Vereinbarung **`EMPTY`** bei **`Zeitraum`**?

(c) Welche **Attribute** sind optional?

(d) Wie lässt sich in der DTD erzwingen, dass der Attributwert zu Datum dem Datumsformat entspricht?

Aufgabe 3:

Ein XML-Schema zum Datenangebot-Dokument sieht wie folgt aus.

Füllen Sie die Lücken. Orientieren Sie sich bei den gewünschten Angaben an der DTD aus Aufgabe 2.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<xsd:element name="Datenangebot" type="DatenangebotT"/>

<xsd:complexType name="DatenangebotT">
  <xsd:sequence>
    <xsd:element name="Anbieter" type="AnbieterT"/>
    <xsd:element name="Volumen" type="VolT"/>
    <xsd:element name="Preis" type="PreisT"/>
  </xsd:sequence>
  <xsd:attribute name="Datum" type="xsd:string" use="___________"/>
  <xsd:attribute name="DatumsFormat" type="xsd:string"
      fixed="DD.MM.YYYY"/>
</xsd:complexType>

<xsd:complexType name="AnbieterT">
  <xsd:sequence>
    <xsd:element name="Kontakt" type="xsd:string"/>
    <xsd:element name="Bemerkung" type="xsd:string"
        minOccurs="_____________" maxOccurs="__________________"/>
  </xsd:sequence>
  <xsd:attribute name="Geschlecht" default="kA">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="M"/>
        <xsd:enumeration value="F"/>
        <xsd:enumeration value="kA"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="Nationalitaet" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="VolT">
  <xsd:sequence>
    <xsd:element name="AnzahlKunden" type="AKundenT"/>
    <xsd:element name="AnzahlSaetze" type="ASaetzeT"
        maxOccurs="unbounded"/>
    <xsd:element name="Zeitraum" type="ZeitraumT" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ASaetzeT">
  <xsd:simpleContent>
    <xsd:extension base="xsd:positiveInteger">
    <xsd:attribute name="Art" default="_____________________">
```

```
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="Stamm"/>
        <xsd:enumeration value="Umsatz"/>
        <xsd:enumeration value="Schliessfach"/>
        <xsd:enumeration value="mixed"/>
      </xsd:restriction>
    </xsd:simpleType>
    </xsd:attribute>
  </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="ZeitraumT">
  <xsd:attribute name="von" type="xsd:string" use="required"/>
  <xsd:attribute name="bis" type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:complexType name="AKundenT">
  <xsd:simpleContent>
    <xsd:extension base="xsd:positiveInteger">
      <xsd:attribute name="SWIFT-BIC" type="____________________"
            use="required"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="PreisT">









</xsd:complexType>

</xsd:schema>
```

Aufgabe 4:

Zeichnen Sie die Tabelle mit der Überschrift, die das XML-Dokument auf der folgenden Seite mit dem unten stehenden Stylesheet in einem der üblichen Browser erzeugt? **Hinweis**: Die Funktion **`normalize-space`** wird gebraucht, um „echten Textinhalt“ von „leerem Text“ (nichts oder nur Leerzeichen, Tabulatoren und Zeilenumbrüche) zu unterscheiden.

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet version='1.0'
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
    <head><title>Datenangebot</title></head>
    <body>
      <h1>Schnellübersicht Angebot vom
        <xsl:value-of select="Datenangebot/@Datum"/>
      </h1>
      <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>

<xsl:template match="Datenangebot">
  <table border="1" cellspacing="0" cellpadding="10">
    <tr>
      <th>Merkmal</th><th>Wert</th><th>Sonstiges</th>
    </tr>
    <xsl:apply-templates/>
  </table>
</xsl:template>

<xsl:template match="*">
  <tr>
    <td><xsl:value-of select="name()"/></td>
    <xsl:choose>
      <xsl:when test="normalize-space(./text()) != ''">
        <td><xsl:value-of select="text()"/></td>
      </xsl:when>
      <xsl:otherwise>
        <td>leer</td>
      </xsl:otherwise>
    </xsl:choose>
    <td>
      <xsl:for-each select="@*">
        <xsl:value-of select="name()"/>
        <xsl:text>: </xsl:text>
        <xsl:value-of select="."/>
        <br/>
      </xsl:for-each>
    </td>
  </tr>
  <xsl:apply-templates select="*"/>
</xsl:template>
</xsl:stylesheet>
```

Dieses Dokument soll dargestellt werden.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="HehlauStyle.xsl" ?>
<Datenangebot Datum="25.02.2010">
  <A a="1">
    <D>2</D>
  </A>
  <B>
    <E b="3">4</E>
    <F c="5">6</F>
    <F d="7">8</F>
    <G e="9" f="10"/>
  </B>
  <C g="11">12</C>
</Datenangebot>
```

Ausgabe:

Aufgabe 5:

Die folgende XQuery soll auf Ihrem Dokument **`Hehlau.xml`** aus Aufgabe 1 laufen. Geben Sie das Ergebnis an!

Hinweis: **fn:round** rundet eine Gleitkommazahl auf eine ganze Zahl. Die Multiplikation und folgende Division mit 100 wird benötigt, um maximal zwei Nachkommastellen (Euro und Cent) für den Preis pro Kunde (PPK) und den Preis pro Datensatz (PPD) im Ergebnis zu haben.

```
let $doc := fn:doc("Hehlau.xml")
let $preis := $doc/Datenangebot/Preis
let $vol := $doc/Datenangebot/Volumen
return
  <PreisLeistung>
    <PPK in="{$preis/@Waehrung}">
    {
      fn:round(($preis div $vol/AnzahlKunden)*100) div 100
    }
    </PPK>
    <PPD in="{$preis/@Waehrung}">
    {
      fn:round(($preis div fn:sum($vol/AnzahlSaetze))*100) div 100
    }
    </PPD>
  </PreisLeistung>
```

Ergebnis:

Aufgabe 6:

Gegeben sei die folgende Datenbanktabelle mit Namen **KONTEN**.

KUNDENNAME	LAND	KONTONR	KNTSTAND
Mustermann	D	4711	200000
Musterfrau	D	4712	5000
Bürger	CH	7411	500000
Steuerfrau	D	1234	300000

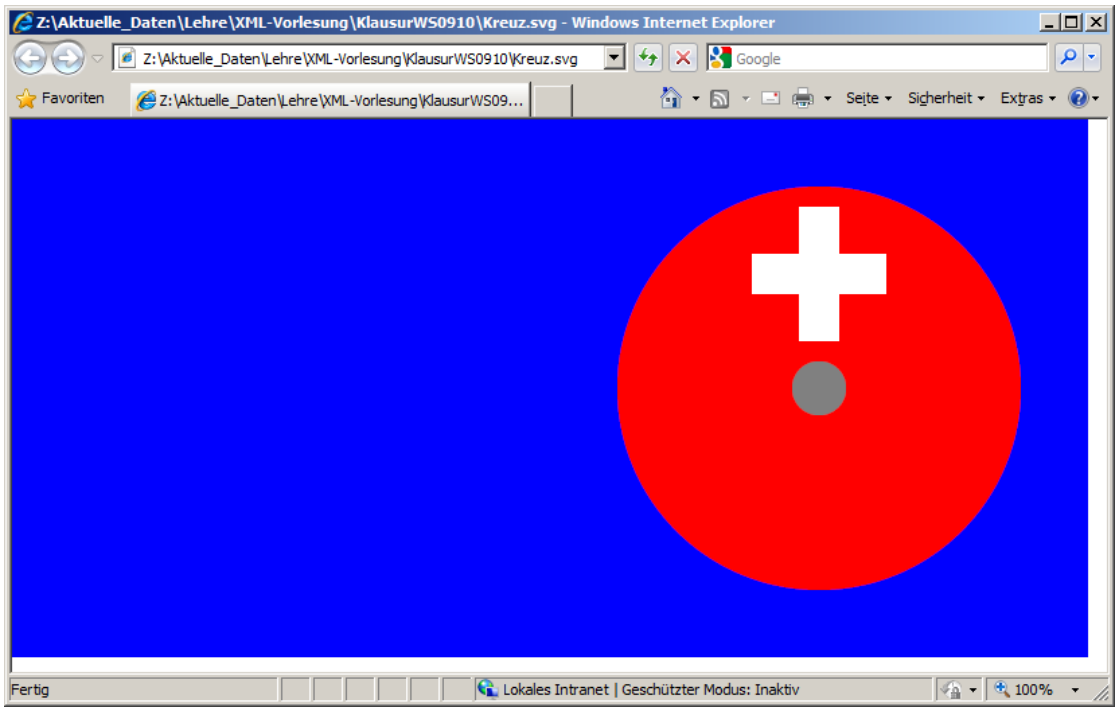
Geben Sie das Ergebnis der untenstehenden SQL/XML-Abfrage an!

```
SELECT XMLELEMENT (
   NAME "Konto",
   XMLELEMENT (NAME "Kunde",
      XMLATTRIBUTES (LAND AS "HLand"),
      KUNDENNAME),
   XMLELEMENT (NAME "Kontonummer",KONTONR),
   XMLELEMENT (NAME "Kontostand",KNTSTAND)
)
FROM KONTEN
WHERE LAND = 'D' AND KNTSTAND > 100000;
```

Ergebnis:

Aufgabe 7:

Die Bundesversammlung der Eidgenossen hat 1889 für das Schweizerkreuz festgelegt, dass das Verhältnis von Breite zu Gesamtlänge des Kreuzbalkens 6:20 beträgt. Tragen Sie die fehlenden Koordinaten ein, damit am Ende der Animation das unten gezeigte Bild entsteht!




```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg">
<rect x="0" y="0" width="800" height="400" fill="blue"/>
<circle r="150" fill="red" cy="200">
  <animate attributeName="cx" from="200" to="600" dur="5s"
    fill="freeze"/>
</circle>
<circle r="20" fill="grey" cy="200">
  <animate attributeName="cx" from="200" to="_________" dur="5s"
    fill="freeze"/>
</circle>

<rect x="_____" y="_____" width="_____" height="100" fill="white"/>
<rect x="_____" y="100" width="_____" height="_____" fill="white"/>
</svg>
```

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

MUSTERLÖSUNG

Nachname: Vorname:

Matr.Nr.: Studiengang:

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 120 Minuten.

Aufgabe	**Punkte max.**	**Punkte erreicht**
1	6	
2	2 + 2 + 2 + 2	
3	4 + 4	
4	8	
5	6	
6	6	
7	6	
Summe	**48**	

Aufgabe 1:

Derzeit werden den Finanzbehörden von verschiedenen Seiten Datenträger mit gestohlenen Bankdaten angeboten. Jede Datensammlung hat ihr eigenes Format. Wir schlagen vor, mit einem einheitlichen Dokument *Hehlau.xml*, das auf jeden Datenträger gehört, Ordnung in dieses Chaos zu bringen. Eine DTD dafür finden Sie in Aufgabe 2.

Geben Sie hier ein zur DTD passendes XML-Dokument für die folgende Datensammlung an.

Das Datenangebot ist vom 25.02.2010 und kommt von einem männlichen schweizer Anbieter (Nationalitätsangabe CH) mit einer Preisvorstellung von 250000 €. Der Kontakt erfolgt über `Speedy4711@taxml.ky`. Als Datenvolumen will er 5000 Kunden liefern, der SWIFT-Bankidentifikationskode lautet `UBSWCHZH80A`, die Anzahl der Datensätze seien 50000 Stammdaten und 200000 Umsatzdaten aus dem Zeitraum 01/2008 bis 12/2009.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Datenangebot SYSTEM "Hehlau.dtd">
<Datenangebot Datum="25.02.2010" DatumsFormat="DD.MM.YYYY">
  <Anbieter Geschlecht="M" Nationalitaet="CH">
    <Kontakt>Speedy4711@taxml.ky</Kontakt>
  </Anbieter>
  <Volumen>
    <AnzahlKunden SWIFT-BIC="UBSWCHZH80A">5000</AnzahlKunden>
    <AnzahlSaetze Art="Stamm">50000</AnzahlSaetze>
    <AnzahlSaetze Art="Umsatz">200000</AnzahlSaetze>
    <Zeitraum von="01/2008" bis="12/2009"/>
  </Volumen>
  <Preis Waehrung="EUR">250000</Preis>
</Datenangebot>
```

Aufgabe 2:

Eine DTD für das Datenangebot aus Aufgabe 1 sieht wie folgt aus.

```
<!-- DTD fuer die DatenCD -->
<!ELEMENT Datenangebot (Anbieter, Volumen, Preis)>
<!ATTLIST Datenangebot Datum CDATA #REQUIRED
                       DatumsFormat CDATA #FIXED "DD.MM.YYYY">

<!ELEMENT Anbieter (Kontakt, Bemerkung*)>
<!ATTLIST Anbieter Geschlecht ( M | F | kA ) "kA"
                   Nationalitaet CDATA #IMPLIED>

<!ELEMENT Kontakt (#PCDATA)>

<!ELEMENT Bemerkung (#PCDATA)>

<!ELEMENT Volumen (AnzahlKunden, AnzahlSaetze+, Zeitraum?)>

<!ELEMENT AnzahlKunden (#PCDATA)>
<!ATTLIST AnzahlKunden SWIFT-BIC CDATA #REQUIRED>

<!ELEMENT AnzahlSaetze (#PCDATA)>
<!ATTLIST AnzahlSaetze Art ( Stamm | Umsatz | Schliessfach | mixed )
            "mixed">

<!ELEMENT Zeitraum EMPTY>
<!ATTLIST Zeitraum von CDATA #REQUIRED
                   bis CDATA #REQUIRED>

<!ELEMENT Preis (#PCDATA)>
<!ATTLIST Preis Waehrung ( EUR | USD | SFR ) "USD">
```

(a) Was bewirkt das Fragezeichen bei der Angabe des Elements **`Zeitraum?`** in **`Volumen`**?

Zeitraum kann weggelassen werden und darf höchstens einmal vorkommen.

(b) Was bewirkt die Vereinbarung **`EMPTY`** bei **`Zeitraum`**?

Zeitraum muss ein leeres Element sein.

(c) Welche **Attribute** sind optional?

DatumsFormat, Geschlecht, Nationalitaet, Art, Waehrung

(d) Wie lässt sich in der DTD erzwingen, dass der Attributwert zu Datum dem Datumsformat entspricht?

geht nicht

Aufgabe 3:

Ein XML-Schema zum Datenangebot-Dokument sieht wie folgt aus.

Füllen Sie die Lücken. Orientieren Sie sich bei den gewünschten Angaben an der DTD aus Aufgabe 2.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<xsd:element name="Datenangebot" type="DatenangebotT"/>

<xsd:complexType name="DatenangebotT">
  <xsd:sequence>
    <xsd:element name="Anbieter" type="AnbieterT"/>
    <xsd:element name="Volumen" type="VolT"/>
    <xsd:element name="Preis" type="PreisT"/>
  </xsd:sequence>
  <xsd:attribute name="Datum" type="xsd:string" use="required"/>
  <xsd:attribute name="DatumsFormat" type="xsd:string"
      fixed="DD.MM.YYYY"/>
</xsd:complexType>

<xsd:complexType name="AnbieterT">
  <xsd:sequence>
    <xsd:element name="Kontakt" type="xsd:string"/>
    <xsd:element name="Bemerkung" type="xsd:string"
        minOccurs="_____0_______" maxOccurs="___unbounded___"/>
  </xsd:sequence>
  <xsd:attribute name="Geschlecht" default="kA">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="M"/>
        <xsd:enumeration value="F"/>
        <xsd:enumeration value="kA"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="Nationalitaet" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="VolT">
  <xsd:sequence>
    <xsd:element name="AnzahlKunden" type="AKundenT"/>
    <xsd:element name="AnzahlSaetze" type="ASaetzeT"
        maxOccurs="unbounded"/>
    <xsd:element name="Zeitraum" type="ZeitraumT" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ASaetzeT">
  <xsd:simpleContent>
    <xsd:extension base="xsd:positiveInteger">
    <xsd:attribute name="Art" default="_____mixed_____">
```

```
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="Stamm"/>
        <xsd:enumeration value="Umsatz"/>
        <xsd:enumeration value="Schliessfach"/>
        <xsd:enumeration value="mixed"/>
      </xsd:restriction>
    </xsd:simpleType>
    </xsd:attribute>
  </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="ZeitraumT">
  <xsd:attribute name="von" type="xsd:string" use="required"/>
  <xsd:attribute name="bis" type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:complexType name="AKundenT">
  <xsd:simpleContent>
    <xsd:extension base="xsd:positiveInteger">
      <xsd:attribute name="SWIFT-BIC" type="xsd:string"
            use="required"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="PreisT">
  <xsd:simpleContent>
    <xsd:extension base="xsd:positiveInteger">
      <xsd:attribute name="Waehrung" default="USD">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="EUR"/>
            <xsd:enumeration value="USD"/>
            <xsd:enumeration value="SFR"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

</xsd:schema>
```

Aufgabe 4:

Zeichnen Sie die Tabelle mit der Überschrift, die das XML-Dokument auf der folgenden Seite mit dem unten stehenden Stylesheet in einem der üblichen Browser erzeugt? **Hinweis**: Die Funktion **`normalize-space`** wird gebraucht, um „echten Textinhalt“ von „leerem Text“ (nichts oder nur Leerzeichen, Tabulatoren und Zeilenumbrüche) zu unterscheiden.

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet version='1.0'
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
    <head><title>Datenangebot</title></head>
    <body>
      <h1>Schnellübersicht Angebot vom
        <xsl:value-of select="Datenangebot/@Datum"/>
      </h1>
      <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>

<xsl:template match="Datenangebot">
  <table border="1" cellspacing="0" cellpadding="10">
    <tr>
      <th>Merkmal</th><th>Wert</th><th>Sonstiges</th>
    </tr>
    <xsl:apply-templates/>
  </table>
</xsl:template>

<xsl:template match="*">
  <tr>
    <td><xsl:value-of select="name()"/></td>
    <xsl:choose>
      <xsl:when test="normalize-space(./text()) != ''">
        <td><xsl:value-of select="text()"/></td>
      </xsl:when>
      <xsl:otherwise>
        <td>leer</td>
      </xsl:otherwise>
    </xsl:choose>
    <td>
      <xsl:for-each select="@*">
        <xsl:value-of select="name()"/>
        <xsl:text>: </xsl:text>
        <xsl:value-of select="."/>
        <br/>
      </xsl:for-each>
    </td>
  </tr>
  <xsl:apply-templates select="*"/>
</xsl:template>
</xsl:stylesheet>
```

Dieses Dokument soll dargestellt werden.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="HehlauStyle.xsl" ?>
<Datenangebot Datum="25.02.2010">
  <A a="1">
    <D>2</D>
  </A>
  <B>
    <E b="3">4</E>
    <F c="5">6</F>
    <F d="7">8</F>
    <G e="9" f="10"/>
  </B>
  <C g="11">12</C>
</Datenangebot>
```

Ausgabe:

Aufgabe 5:

Die folgende XQuery soll auf Ihrem Dokument **`Hehlau.xml`** aus Aufgabe 1 laufen. Geben Sie das Ergebnis an!

Hinweis: **fn:round** rundet eine Gleitkommazahl auf eine ganze Zahl. Die Multiplikation und folgende Division mit 100 wird benötigt, um maximal zwei Nachkommastellen (Euro und Cent) für den Preis pro Kunde (PPK) und den Preis pro Datensatz (PPD) im Ergebnis zu haben.

```
let $doc := fn:doc("Hehlau.xml")
let $preis := $doc/Datenangebot/Preis
let $vol := $doc/Datenangebot/Volumen
return
  <PreisLeistung>
    <PPK in="{$preis/@Waehrung}">
    {
      fn:round(($preis div $vol/AnzahlKunden)*100) div 100
    }
    </PPK>
    <PPD in="{$preis/@Waehrung}">
    {
      fn:round(($preis div fn:sum($vol/AnzahlSaetze))*100) div 100
    }
    </PPD>
  </PreisLeistung>
```

Ergebnis:

```
<PreisLeistung>
  <PPK in="EUR">50</PPK>
  <PPD in="EUR">1</PPD>
</PreisLeistung>
```

Aufgabe 6:

Gegeben sei die folgende Datenbanktabelle mit Namen **KONTEN**.

KUNDENNAME	LAND	KONTONR	KNTSTAND
Mustermann	D	4711	200000
Musterfrau	D	4712	5000
Bürger	CH	7411	500000
Steuerfrau	D	1234	300000

Geben Sie das Ergebnis der untenstehenden SQL/XML-Abfrage an!

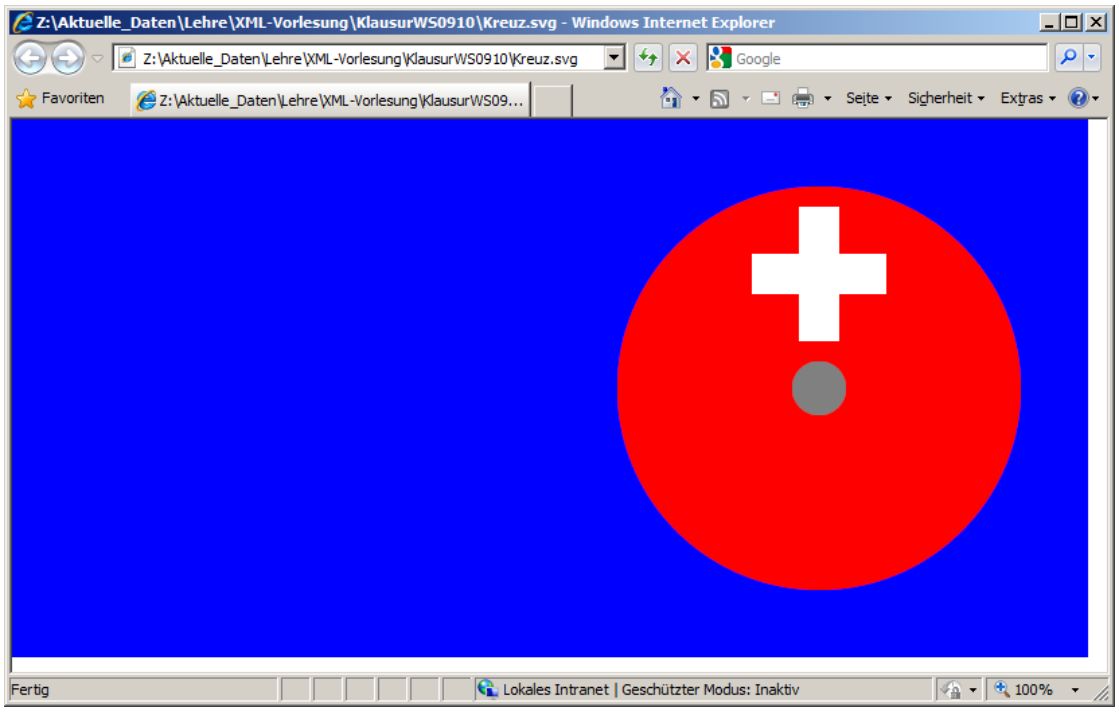
```
SELECT XMLELEMENT (
   NAME "Konto",
   XMLELEMENT (NAME "Kunde",
      XMLATTRIBUTES (LAND AS "HLand"),
      KUNDENNAME),
   XMLELEMENT (NAME "Kontonummer",KONTONR),
   XMLELEMENT (NAME "Kontostand",KNTSTAND)
)
FROM KONTEN
WHERE LAND = 'D' AND KNTSTAND > 100000;
```

Ergebnis:

```
<Konto>
  <Kunde HLand="D">Mustermann</Kunde>
  <Kontonummer>4711</Kontonummer>
  <Kontostand>200000</Kontostand>
</Konto>
<Konto>
  <Kunde HLand="D">Steuerfrau</Kunde>
  <Kontonummer>1234</Kontonummer>
  <Kontostand>300000</Kontostand>
</Konto>
```

Aufgabe 7:

Die Bundesversammlung der Eidgenossen hat 1889 für das Schweizerkreuz festgelegt, dass das Verhältnis von Breite zu Gesamtlänge des Kreuzbalkens 6:20 beträgt. Tragen Sie die fehlenden Koordinaten ein, damit am Ende der Animation das unten gezeigte Bild entsteht!




```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg">
<rect x="0" y="0" width="800" height="400" fill="blue"/>
<circle r="150" fill="red" cy="200">
  <animate attributeName="cx" from="200" to="600" dur="5s"
    fill="freeze"/>
</circle>
<circle r="20" fill="grey" cy="200">
  <animate attributeName="cx" from="200" to="600" dur="5s"
    fill="freeze"/>
</circle>

<rect x="585" y="65" width="30" height="100" fill="white"/>
<rect x="550" y="100" width="100" height="30" fill="white"/>
</svg>
```

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

Nachname: Vorname:

Matr.Nr.: Studiengang:

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 120 Minuten.

Aufgabe	**Punkte max.**	**Punkte erreicht**
1	2 + 2 + 2	
2	2 + 2 + 2 + 2 + 2	
3	8	
4	7	
5	5	
6	6	
7	6	
Summe	**48**	

Aufgabe 1:

Wir arbeiten hier nochmals die Bundespräsidentenwahl durch die 14. Bundesversammlung am 30. Juni 2010 auf, weil natürlich die Ergebnisse der drei Wahlgänge in eine ordentliche XML-Form gebracht werden müssen. Eine DTD für die von uns vorgeschlagene Form finden Sie in Aufgabe 2. Im Dokument unten haben wir den mittleren Teil weggelassen.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Wahl SYSTEM "Wahl.dtd">
<Wahl Datum="30.06.2010" Stimmberechtigte="1244">
  <Wahlgang Nr="1" abgegebene Stimmen="1242">
    <Stimmen Art="ungueltig">1</Stimmen>
    <Stimmen Art="Enthaltung">13</Stimmen>
    <Stimmen Art="fuer" Kandidat="Christian Wulff">600</Stimmen>
    <Stimmen Art="fuer" Kandidat="Joachim Gauck">499</Stimmen>
    <Stimmen Art="fuer" Kandidat="Luc Jochimsen">126</Stimmen>
    <Stimmen Art="fuer" Kandidat="Frank Rennicke">3</Stimmen>
    <Bemerkung>Absolute Mehrheit notwendig</Bemerkung>
  </Wahlgang>
  <Wahlgang Nr="2" abgegebene Stimmen="1239">
    ...
  </Wahlgang>
  <Wahlgang Nr="3" abgegebene Stimmen="1242">
    <Stimmen Art="ungueltig">2</Stimmen>
    <Stimmen Art="Enthaltung">121</Stimmen>
    <Stimmen Art="fuer" Kandidat="Christian Wulff">625</Stimmen>
    <Stimmen Art="fuer" Kandidat="Joachim Gauck">494</Stimmen>
    <Bemerkung>Relative Mehrheit ausreichend</Bemerkung>
  </Wahlgang>
  <Ergebnis gewaehlt="Christian Wulff"/>
</Wahl>
```

(a) Gibt es Elemente mit „mixed content“? Wenn ja, welche?

(b) Welches Attribut hat bzw. welche Attribute haben im Dokument eine ungültige Syntax?

(c) Welches Element ist leer bzw. welche sind leer?

Aufgabe 2:

Eine DTD für die Wahl zum Bundespräsidenten aus Aufgabe 1 sieht wie folgt aus.

```
<!-- DTD fuer die Bundespraesidentenwahl -->
<!ELEMENT Wahl (Wahlgang+, Ergebnis)>
<!ATTLIST Wahl Datum CDATA #REQUIRED
               Stimmberechtigte CDATA #REQUIRED>

<!ELEMENT Wahlgang (Stimmen+, Bemerkung?)>
<!ATTLIST Wahlgang Nr CDATA #REQUIRED
                   abgegebeneStimmen CDATA #REQUIRED>

<!ELEMENT Bemerkung (#PCDATA)>

<!ELEMENT Stimmen (#PCDATA)>
<!ATTLIST Stimmen Art ( ungueltig | Enthaltung | fuer ) "fuer"
                  Kandidat CDATA #IMPLIED>

<!ELEMENT Ergebnis EMPTY>
<!ATTLIST Ergebnis gewaehlt CDATA #REQUIRED>
```

(a) Was bewirkt das Fragezeichen **`?`** bei der Angabe des Elements **`Bemerkung`** in **`Wahlgang`**?

(b) Was muss beim Element **`Bemerkung`** geändert werden, damit es auch leer sein darf?

(c) Welche **Attribute** sind optional?

(d) Wie erreicht man in der DTD, dass das Attribut **`Kandidat`** nur erscheint, wenn der Attributwert **`Art`** den Wert **`"fuer"`** hat?

(e) Könnte bzw. sollte man in **`Wahlgang`** beim Attribut **`Nr`** statt **`CDATA`** besser **`INTEGER`** verwenden?

Aufgabe 3:

Ein XML-Schema zum Wahl-Dokument sieht wie folgt aus. Ergänzen Sie den Teil zu **`WahlgangT`**. Orientieren Sie sich bei den gewünschten Angaben an der DTD aus Aufgabe 2.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<xsd:element name="Wahl" type="WahlT"/>

<xsd:complexType name="WahlT">
  <xsd:sequence>
    <xsd:element name="Wahlgang" type="WahlgangT"
        maxOccurs="unbounded"/>
    <xsd:element name="Ergebnis">
      <xsd:complexType>
        <xsd:attribute name="gewaehlt" type="xsd:string"
              use="required"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
  <xsd:attribute name="Datum" type="xsd:string" use="required"/>
  <xsd:attribute name="Stimmberechtigte"
        type="xsd:positiveInteger" use="required"/>
</xsd:complexType>

<xsd:complexType name="WahlgangT">
```

Fortsetzung auf folgender Seite möglich

```
</xsd:complexType>

<xsd:complexType name="StimmenT">
  <xsd:simpleContent>
    <xsd:extension base="xsd:positiveInteger">
      <xsd:attribute name="Art" default="fuer">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="ungueltig"/>
            <xsd:enumeration value="Enthaltung"/>
            <xsd:enumeration value="fuer"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="Kandidat" type="xsd:string"
          use="optional"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

</xsd:schema>
```

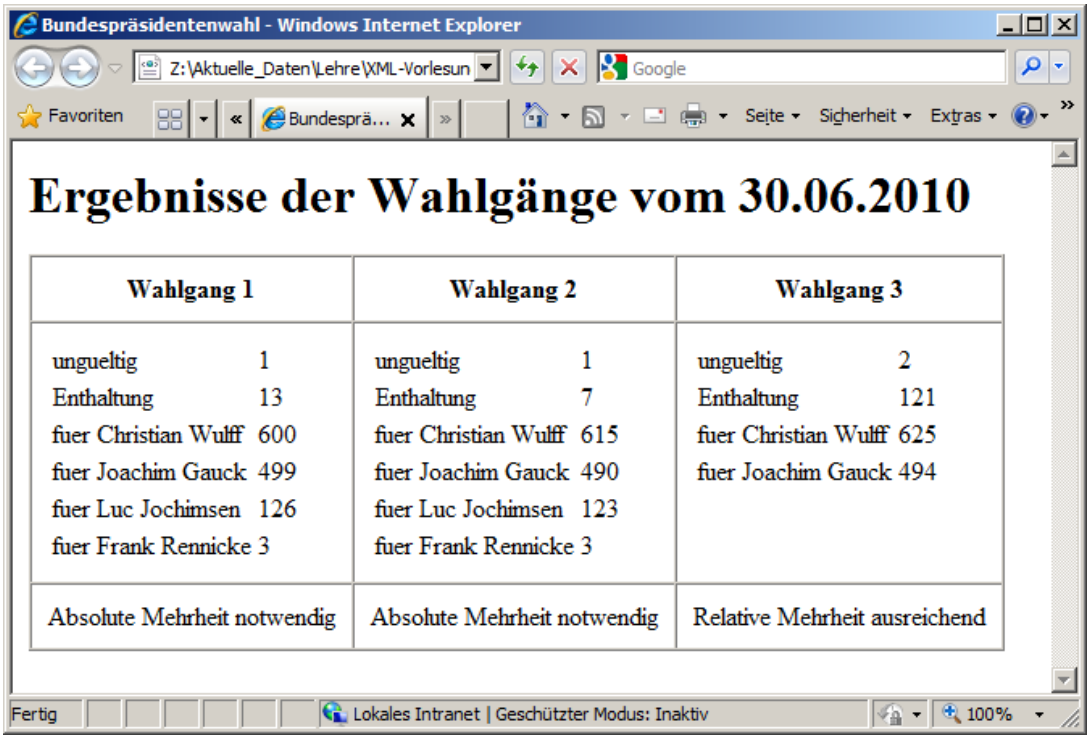
Aufgabe 4:

Füllen Sie die Lücken im folgenden Stylesheet. Es soll das Dokument aus Aufgabe 1 dargestellt werden. Die gewünschte Ausgabe wird unten gezeigt.

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet version='1.0'
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
    <head><title>Bundespräsidentenwahl</title></head>
    <body>
      <h1>Ergebnisse der Wahlgänge vom
      <xsl:value-of select="_____________________________"/>
      </h1>

      ____________________________________________
    </body>
  </html>
</xsl:template>

<xsl:template match="Wahl">
  <table border="1" cellspacing="0" cellpadding="10">
    <tr>
      <xsl:for-each select="Wahlgang">
        <th>Wahlgang <xsl:__________________ select="@Nr"/></th>
      </xsl:for-each>
    </tr>
    <tr valign="top">
      <xsl:for-each select="Wahlgang">
        <td>
          <table>
            <xsl:apply-templates select="____________________"/>
          </table>
        </td>
      </xsl:for-each>
    </tr>
    <tr>
      <xsl:for-each select="Wahlgang">
        <td><xsl:value-of select="Bemerkung"/></td>
      </xsl:for-each>
    </tr>
```

```
  </table>
</xsl:template>

<xsl:template match="Stimmen">
  <tr>
    <td>
      <xsl:value-of select="@Art"/>
      <xsl:if test="_____________ = '_________________'">
        <xsl:text> </xsl:text>
        <xsl:value-of select="@Kandidat"/>
      </xsl:if>
    </td>
    <td>
      <xsl:value-of select="________"/>
    </td>
  </tr>
</xsl:template>
</xsl:stylesheet>
```

Ausgabe:



Aufgabe 5:

Die folgende XQuery soll auf dem Dokument **Wahl.xml** aus Aufgabe 1 laufen und das unten gezeigte Ergebnis liefern. Füllen Sie die Lücken in der XQuery!

Hinweis: **fn:round** rundet eine Gleitkommazahl auf eine ganze Zahl. Die Multiplikation mit 1000 und folgende Division durch 10 wird benötigt, um maximal eine Nachkommastellen für die Prozentangabe (Stimmen/abgegebene Stimmen insgesamt) je Wahlgang im Ergebnis zu haben.

```
let $doc := fn:doc("Wahl.xml")
return
  <Auszaehlung>
  {
    for $wgang in $doc/Wahl/Wahlgang
    return
      <Wahlgang Nummer="{_____________________________________}">
      {
        for $stimmen in $wgang/_____________________________
        return
          <Stimmen>
          {
            fn:concat($stimmen/__________________,' ',$stimmen/@Kandidat,' ',
            _____________________________,' (',
            fn:round(($stimmen div $wgang/_____________________________)*1000)
            div 10,'%)')
          }
          </Stimmen>
      }
      </Wahlgang>
  }
  </Auszaehlung>
```

Ergebnis:

```
<Auszaehlung>
  <Wahlgang Nummer="1">
    ...
  </Wahlgang>
  <Wahlgang Nummer="2">
    ...
  </Wahlgang>
  <Wahlgang Nummer="3">
    <Stimmen>ungueltig  2 (0.2%)</Stimmen>
    <Stimmen>Enthaltung  121 (9.7%)</Stimmen>
    <Stimmen>fuer Christian Wulff 625 (50.3%)</Stimmen>
    <Stimmen>fuer Joachim Gauck 494 (39.8%)</Stimmen>
  </Wahlgang>
</Auszaehlung>
```

Aufgabe 6:

Gegeben sei die folgende Datenbanktabelle mit Namen **WAHLGAENGE**.

KANDIDAT	VON	WAHLGANG	STIMMEN	PROZENT
Joachim Gauck	SPD/Grüne	1	499	40,1
Christian Wulff	CDU/CSU/FDP	1	600	48,2
Luc Jochimsen	Die Linke	1	126	10,1
Luc Jochimsen	Die Linke	2	123	9,9
Joachim Gauck	SPD/Grüne	2	490	39,4
Joachim Gauck	SPD/Grüne	3	494	39,8
Christian Wulff	CDU/CSU/FDP	2	615	49,4
Christian Wulff	CDU/CSU/FDP	3	625	50,3

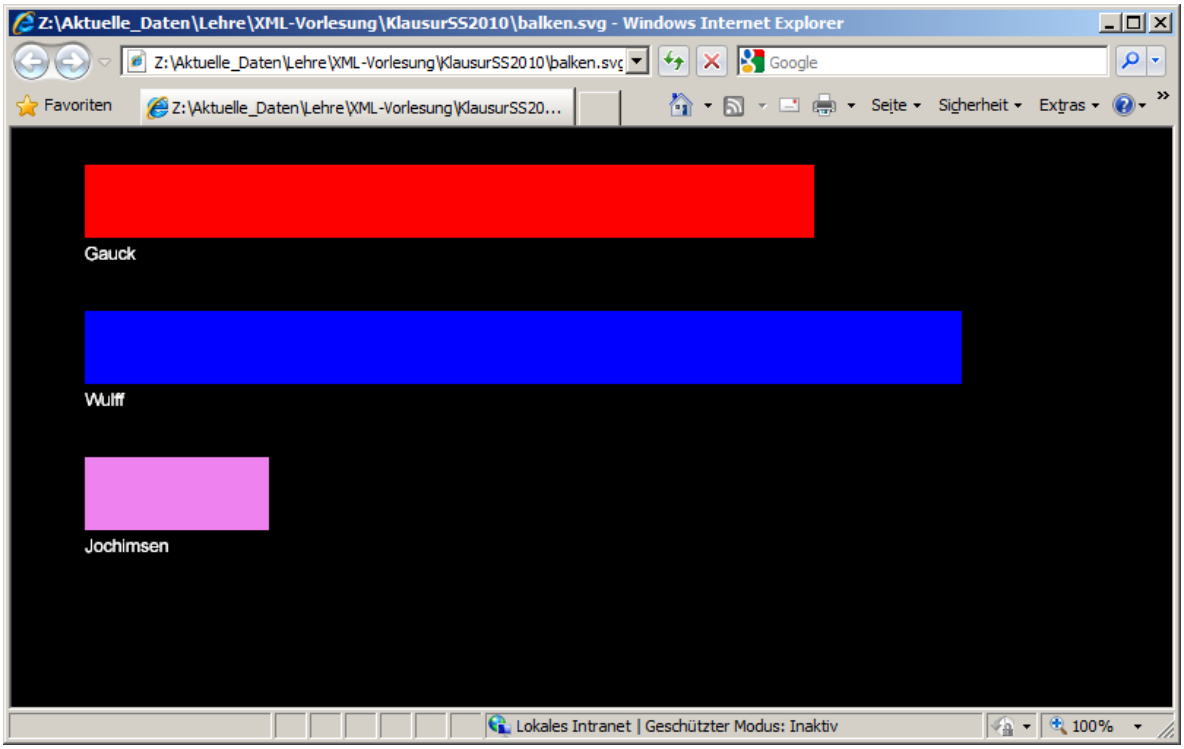
Geben Sie das Ergebnis der untenstehenden SQL/XML-Abfrage an und beachten Sie das **ORDER BY** ... **DESC**!

```
SELECT XMLELEMENT (
  NAME "Ergebnis",
    XMLELEMENT (NAME "Kandidat",
      XMLATTRIBUTES (VON AS "Parteien"),
      KANDIDAT),
    XMLELEMENT (NAME "Stimmen",
      XMLATTRIBUTES (WAHLGANG),
      STIMMEN)
)
FROM WAHLGAENGE
WHERE WAHLGANG = 3
ORDER BY STIMMEN DESC;
```

Ergebnis:

Aufgabe 7:

Wahlausgänge werden gern mit wachsenden Balken dargestellt. Füllen Sie die Lücken im SVG-Dokument unten, damit am Ende der Animation das folgende Bild angezeigt bleibt! Orientieren Sie sich an den Stimmen aus dem 1. Wahlgang aus Aufgabe 1.




```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg">
<________ x="0" y="0" width="800" height="400" fill="black"/>
<rect x="50" y="25" height="50" fill="red">
  <animate attributeName="___________" from="0" to="499" dur="5s"
    fill="_____________"/>
</rect>
<text x="50" y="90" fill="white" text-anchor="left">Gauck</text>
<rect x="50" y="125" height="50" fill="blue">
  <animate attributeName="___________" from="0" to="_____" dur="5s"
    fill="_____________"/>
</rect>
<text x="50" y="190" fill="white" text-anchor="left">Wulff</text>
<rect x="50" y="225" height="50" fill="violet">
  <animate attributeName="___________" from="0" to="126" dur="5s"
    fill="_____________"/>
</rect>
<text x="50" y="___" fill="white" text-anchor="left">Jochimsen</text>
</svg>
```

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

Nachname: Vorname:

Mat.Nr.: Studiengang:

MUSTERLÖSUNG

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 120 Minuten.

Aufgabe	Punkte max.	Punkte erreicht
1	2 + 2 + 2	
2	2 + 2 + 2 + 2 + 2	
3	8	
4	7	
5	5	
6	6	
7	6	
Summe	**48**	

Aufgabe 1:

Wir arbeiten hier nochmals die Bundespräsidentenwahl durch die 14. Bundesversammlung am 30. Juni 2010 auf, weil natürlich die Ergebnisse der drei Wahlgänge in eine ordentliche XML-Form gebracht werden müssen. Eine DTD für die von uns vorgeschlagene Form finden Sie in Aufgabe 2. Im Dokument unten haben wir den mittleren Teil weggelassen.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Wahl SYSTEM "Wahl.dtd">
<Wahl Datum="30.06.2010" Stimmberechtigte="1244">
  <Wahlgang Nr="1" abgegebene Stimmen="1242">
    <Stimmen Art="ungueltig">1</Stimmen>
    <Stimmen Art="Enthaltung">13</Stimmen>
    <Stimmen Art="fuer" Kandidat="Christian Wulff">600</Stimmen>
    <Stimmen Art="fuer" Kandidat="Joachim Gauck">499</Stimmen>
    <Stimmen Art="fuer" Kandidat="Luc Jochimsen">126</Stimmen>
    <Stimmen Art="fuer" Kandidat="Frank Rennicke">3</Stimmen>
    <Bemerkung>Absolute Mehrheit notwendig</Bemerkung>
  </Wahlgang>
  <Wahlgang Nr="2" abgegebene Stimmen="1239">
    ...
  </Wahlgang>
  <Wahlgang Nr="3" abgegebene Stimmen="1242">
    <Stimmen Art="ungueltig">2</Stimmen>
    <Stimmen Art="Enthaltung">121</Stimmen>
    <Stimmen Art="fuer" Kandidat="Christian Wulff">625</Stimmen>
    <Stimmen Art="fuer" Kandidat="Joachim Gauck">494</Stimmen>
    <Bemerkung>Relative Mehrheit ausreichend</Bemerkung>
  </Wahlgang>
  <Ergebnis gewaehlt="Christian Wulff"/>
</Wahl>
```

(a) Gibt es Elemente mit „mixed content“? Wenn ja, welche?

keine Elemente

(b) Welches Attribut hat bzw. welche Attribute haben im Dokument eine ungültige Syntax?

„abgegebene Stimmen“

(c) Welches Element ist leer bzw. welche sind leer?

Ergebnis

Aufgabe 2:

Eine DTD für die Wahl zum Bundespräsidenten aus Aufgabe 1 sieht wie folgt aus.

```
<!-- DTD fuer die Bundespraesidentenwahl -->
<!ELEMENT Wahl (Wahlgang+, Ergebnis)>
<!ATTLIST Wahl Datum CDATA #REQUIRED
               Stimmberechtigte CDATA #REQUIRED>

<!ELEMENT Wahlgang (Stimmen+, Bemerkung?)>
<!ATTLIST Wahlgang Nr CDATA #REQUIRED
                   abgegebeneStimmen CDATA #REQUIRED>

<!ELEMENT Bemerkung (#PCDATA)>

<!ELEMENT Stimmen (#PCDATA)>
<!ATTLIST Stimmen Art ( ungueltig | Enthaltung | fuer ) "fuer"
                  Kandidat CDATA #IMPLIED>

<!ELEMENT Ergebnis EMPTY>
<!ATTLIST Ergebnis gewaehlt CDATA #REQUIRED>
```

(a) Was bewirkt das Fragezeichen **?** bei der Angabe des Elements **Bemerkung** in **Wahlgang**?

Das Element Bemerkung kann weggelassen oder genau einmal in Wahlgang verwendet werden.

(b) Was muss beim Element **Bemerkung** geändert werden, damit es auch leer sein darf?

Nichts.

(c) Welche **Attribute** sind optional?

Art, Kandidat

(d) Wie erreicht man in der DTD, dass das Attribut **Kandidat** nur erscheint, wenn der Attributwert **Art** den Wert **"fuer"** hat?

Nicht möglich.

(e) Könnte bzw. sollte man in **Wahlgang** beim Attribut **Nr** statt **CDATA** besser **INTEGER** verwenden?

Nein, nicht möglich.

Aufgabe 3:

Ein XML-Schema zum Wahl-Dokument sieht wie folgt aus. Ergänzen Sie den Teil zu **`WahlgangT`**. Orientieren Sie sich bei den gewünschten Angaben an der DTD aus Aufgabe 2.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<xsd:element name="Wahl" type="WahlT"/>

<xsd:complexType name="WahlT">
  <xsd:sequence>
    <xsd:element name="Wahlgang" type="WahlgangT"
        maxOccurs="unbounded"/>
    <xsd:element name="Ergebnis">
      <xsd:complexType>
        <xsd:attribute name="gewaehlt" type="xsd:string"
              use="required"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
  <xsd:attribute name="Datum" type="xsd:string" use="required"/>
  <xsd:attribute name="Stimmberechtigte"
        type="xsd:positiveInteger" use="required"/>
</xsd:complexType>

<xsd:complexType name="WahlgangT">
  <xsd:sequence>
    <xsd:element name="Stimmen" type="StimmenT"
      maxOccurs="unbounded"/>
    <xsd:element name="Bemerkung" type="xsd:string"
      minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="Nr" type="xsd:positiveInteger"
    use="required"/>
  <xsd:attribute name="abgegebeneStimmen"
    type="xsd:positiveInteger" use="required"/>
</xsd:complexType>
```

xsd:string auch als richtig gewertet.

Fortsetzung auf folgender Seite möglich

```
<xsd:complexType name="StimmenT">
  <xsd:simpleContent>
    <xsd:extension base="xsd:positiveInteger">
      <xsd:attribute name="Art" default="fuer">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="ungueltig"/>
            <xsd:enumeration value="Enthaltung"/>
            <xsd:enumeration value="fuer"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="Kandidat" type="xsd:string"
          use="optional"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

</xsd:schema>
```

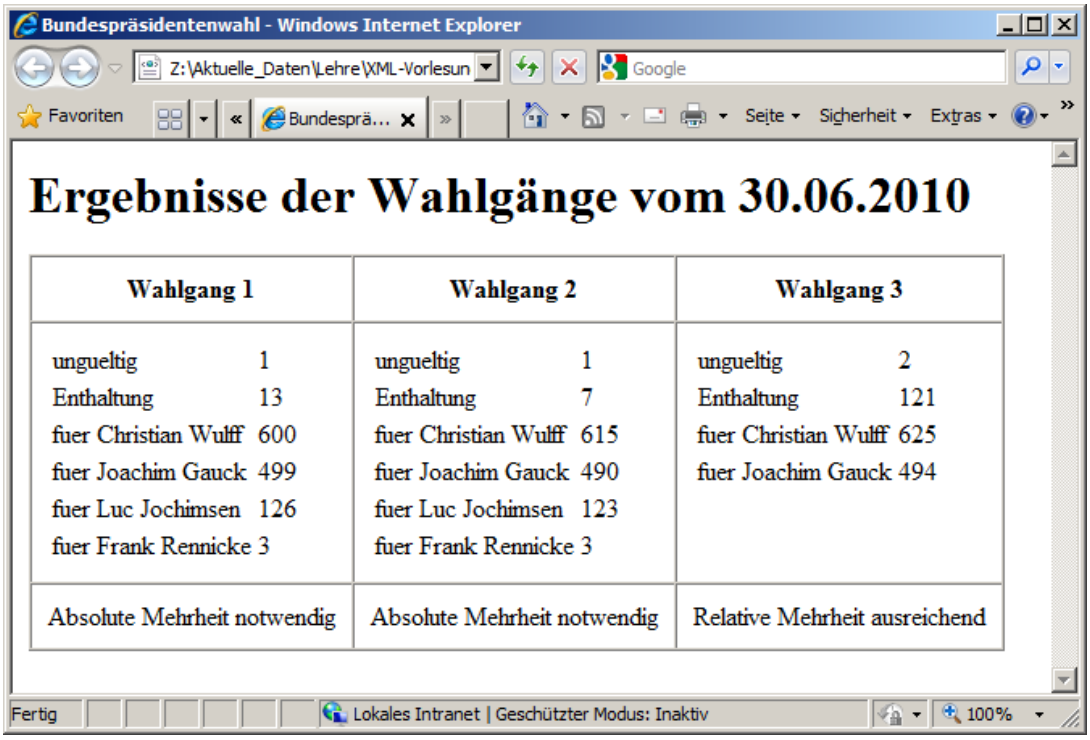
Aufgabe 4:

Füllen Sie die Lücken im folgenden Stylesheet. Es soll das Dokument aus Aufgabe 1 dargestellt werden. Die gewünschte Ausgabe wird unten gezeigt.

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet version='1.0'
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
    <head><title>Bundespräsidentenwahl</title></head>
    <body>
      <h1>Ergebnisse der Wahlgänge vom
      <xsl:value-of select="Wahl/@Datum"/>
      </h1>
        <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>

<xsl:template match="Wahl">
  <table border="1" cellspacing="0" cellpadding="10">
    <tr>
      <xsl:for-each select="Wahlgang">
        <th>Wahlgang <xsl:value-of select="@Nr"/></th>
      </xsl:for-each>
    </tr>
    <tr valign="top">
      <xsl:for-each select="Wahlgang">
        <td>
          <table>
            <xsl:apply-templates select="Stimmen"/>
          </table>
        </td>
      </xsl:for-each>
    </tr>
    <tr>
      <xsl:for-each select="Wahlgang">
        <td><xsl:value-of select="Bemerkung"/></td>
      </xsl:for-each>
    </tr>
```

```
  </table>
</xsl:template>

<xsl:template match="Stimmen">
  <tr>
    <td>
      <xsl:value-of select="@Art"/>
      <xsl:if test="@Art = 'fuer'">
        <xsl:text> </xsl:text>
        <xsl:value-of select="@Kandidat"/>
      </xsl:if>
    </td>
    <td>
      <xsl:value-of select="."/>
    </td>
  </tr>
</xsl:template>
</xsl:stylesheet>
```

Ausgabe:



Aufgabe 5:

Die folgende XQuery soll auf dem Dokument **Wahl.xml** aus Aufgabe 1 laufen und das unten gezeigte Ergebnis liefern. Füllen Sie die Lücken in der XQuery!

Hinweis: **fn:round** rundet eine Gleitkommazahl auf eine ganze Zahl. Die Multiplikation mit 1000 und folgende Division durch 10 wird benötigt, um maximal eine Nachkommastellen für die Prozentangabe (Stimmen/abgegebene Stimmen insgesamt) je Wahlgang im Ergebnis zu haben.

```
let $doc := fn:doc("Wahl.xml")
return
  <Auszaehlung>
  {
    for $wgang in $doc/Wahl/Wahlgang
    return
      <Wahlgang Nummer="{$wgang/@Nr}">
      {
        for $stimmen in $wgang/Stimmen
        return
          <Stimmen>
          {
            fn:concat($stimmen/@Art,' ',$stimmen/@Kandidat,' ',
            $stimmen,' (',
            fn:round(($stimmen div $wgang/@abgegebeneStimmen)*1000)
            div 10,'%)')
          }
          </Stimmen>
      }
      </Wahlgang>
  }
  </Auszaehlung>
```

Ergebnis:

```
<Auszaehlung>
  <Wahlgang Nummer="1">
    ...
  </Wahlgang>
  <Wahlgang Nummer="2">
    ...
  </Wahlgang>
  <Wahlgang Nummer="3">
    <Stimmen>ungueltig  2 (0.2%)</Stimmen>
    <Stimmen>Enthaltung  121 (9.7%)</Stimmen>
    <Stimmen>fuer Christian Wulff 625 (50.3%)</Stimmen>
    <Stimmen>fuer Joachim Gauck 494 (39.8%)</Stimmen>
  </Wahlgang>
</Auszaehlung>
```

Aufgabe 6:

Gegeben sei die folgende Datenbanktabelle mit Namen **WAHLGAENGE**.

KANDIDAT	VON	WAHLGANG	STIMMEN	PROZENT
Joachim Gauck	SPD/Grüne	1	499	40,1
Christian Wulff	CDU/CSU/FDP	1	600	48,2
Luc Jochimsen	Die Linke	1	126	10,1
Luc Jochimsen	Die Linke	2	123	9,9
Joachim Gauck	SPD/Grüne	2	490	39,4
Joachim Gauck	SPD/Grüne	3	494	39,8
Christian Wulff	CDU/CSU/FDP	2	615	49,4
Christian Wulff	CDU/CSU/FDP	3	625	50,3

Geben Sie das Ergebnis der untenstehenden SQL/XML-Abfrage an und beachten Sie das **ORDER BY** ... **DESC**!

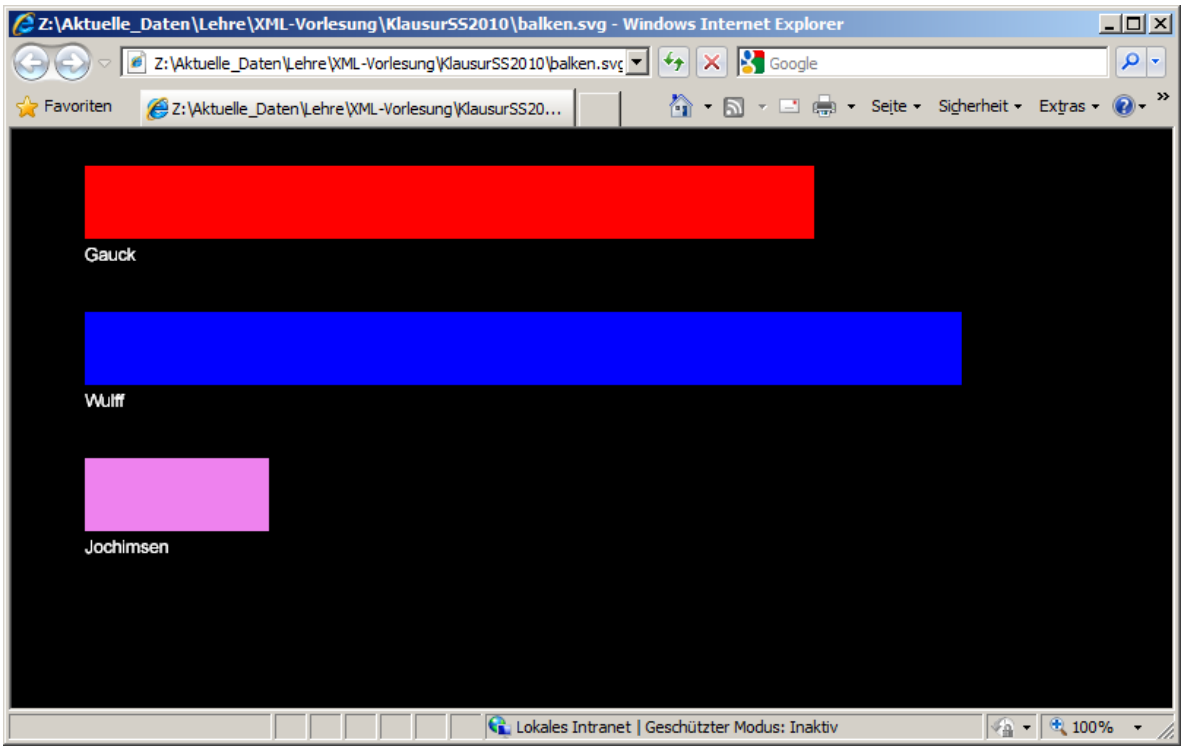
```
SELECT XMLELEMENT (
  NAME "Ergebnis",
    XMLELEMENT (NAME "Kandidat",
      XMLATTRIBUTES (VON AS "Parteien"),
      KANDIDAT),
    XMLELEMENT (NAME "Stimmen",
      XMLATTRIBUTES (WAHLGANG),
      STIMMEN)
)
FROM WAHLGAENGE
WHERE WAHLGANG = 3
ORDER BY STIMMEN DESC;
```

Ergebnis:

```
<Ergebnis>
  <Kandidat Parteien="CDU/CSU/FDP">Christian Wulff</Kandidat>
  <Stimmen WAHLGANG="3">625</Stimmen>
</Ergebnis>
<Ergebnis>
  <Kandidat Parteien="SPD/Grüne">Joachim Gauck</Kandidat>
  <Stimmen WAHLGANG="3">494</Stimmen>
</Ergebnis>
```

Aufgabe 7:

Wahlausgänge werden gern mit wachsenden Balken dargestellt. Füllen Sie die Lücken im SVG-Dokument unten, damit am Ende der Animation das folgende Bild angezeigt bleibt! Orientieren Sie sich an den Stimmen aus dem 1. Wahlgang aus Aufgabe 1.




```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg">
<rect x="0" y="0" width="800" height="400" fill="black"/>
<rect x="50" y="25" height="50" fill="red">
  <animate attributeName="width" from="0" to="499" dur="5s"
    fill="freeze"/>
</rect>
<text x="50" y="90" fill="white" text-anchor="left">Gauck</text>
<rect x="50" y="125" height="50" fill="blue">
  <animate attributeName="width" from="0" to="600" dur="5s"
    fill="freeze"/>
</rect>
<text x="50" y="190" fill="white" text-anchor="left">Wulff</text>
<rect x="50" y="225" height="50" fill="violet">
  <animate attributeName="width" from="0" to="126" dur="5s"
    fill="freeze"/>
</rect>
<text x="50" y="290" fill="white" text-anchor="left">Jochimsen</text>
</svg>
```

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

Nachname: Vorname:

Matr.Nr.: Studiengang:

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 120 Minuten.

Aufgabe	Punkte max.	Punkte erreicht
1	6	
2	2 + 2 + 2 + 2	
3	6	
4	6	
5	6	
6	6	
7	6	
Summe	**44**	

Aufgabe 1:

Die Untermietverhältnisse in einer Wohngemeinschaft (WG) sollen durch ein XML-Dokument **`WG.xml`** festgehalten werden. Das Format des Dokuments wird in der DTD in Aufgabe 2 definiert. Die Daten für die fünf Zimmer der WG kann man der Stylesheet-Darstellung in Aufgabe 4 entnehmen.

Vervollständigen Sie das Dokument **`WG.xml`** einschließlich aller Attribute, aber **nur für Wohnung, Adresse und die ersten zwei Zimmer und deren Unterelemente**!

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="WGStyle.xsl" ?>
<!DOCTYPE Wohnung SYSTEM "WG.dtd">
<Wohnung ...
...
</Wohnung>
```

Aufgabe 2:

Die DTD für die Wohngemeinschaft aus Aufgabe 1 sieht wie folgt aus.

```
<!-- DTD fuer die Wohngemeinschaft -->
<!ELEMENT Wohnung (Adresse, Zimmer+)>
<!ATTLIST Wohnung    Zimmeranzahl CDATA #REQUIRED
                     QMGesamt CDATA #REQUIRED
                     QMAllgemein CDATA #REQUIRED
                     Kaltmiete CDATA #REQUIRED
                     Nebenkosten CDATA #REQUIRED>

<!ELEMENT Adresse (#PCDATA)>

<!ELEMENT Zimmer (UM, Miete)>
<!ATTLIST Zimmer  Nummer ID #REQUIRED
                  QMZimmer CDATA #REQUIRED>

<!ELEMENT UM (#PCDATA)>

<!ELEMENT Miete EMPTY>
<!ATTLIST Miete  KM CDATA #IMPLIED
                 NK CDATA #IMPLIED>
```

(a) Wie kann man leicht erreichen, dass die DTD für eine **`Wohnung`** mindestens zwei **`Zimmer`** verlangt?

(b) Was bewirkt die Vereinbarung **`EMPTY`** bei **`Miete`**?

(c) Welche Attribute und Elemente dürfen weggelassen werden?

(d) Wird verhindert, dass es zwei Zimmer mit gleicher Zimmernummer gibt? Wenn ja, womit erreicht man das; wenn nein, warum leistet diese DTD das nicht?

Aufgabe 3:

Ein XML-Schema zum WG-Dokument sieht wie folgt aus.

Füllen Sie die Lücken. Orientieren Sie sich bei den gewünschten Angaben an der DTD aus Aufgabe 2.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<xsd:element name="_____________________" type="_________________"/>

<xsd:complexType name="WohnungT">
    <xsd:sequence>
        <xsd:element name="Adresse" type="xsd:string"/>
        <xsd:element name="Zimmer" type="ZimmerT"
            minOccurs="1" maxOccurs="10"/>
    </xsd:sequence>
    <xsd:attribute name="Zimmeranzahl" type="xsd:decimal"
            use="___________"/>
    <xsd:attribute name="QMGesamt" type="xsd:decimal"
            use="___________"/>
    <xsd:attribute name="QMAllgemein" type="xsd:decimal"
            use="___________"/>
    <xsd:attribute name="Kaltmiete" type="xsd:decimal"
            use="___________"/>
    <xsd:attribute name="Nebenkosten" type="xsd:decimal"
            use="___________"/>

</xsd:complexType>

<xsd:complexType name="ZimmerT">
    <xsd:sequence>
        <xsd:element name="UM" type="xsd:string"/>
        <xsd:element name="Miete" type="__________________"/>
    </xsd:sequence>
    <xsd:attribute name="__________" type="xsd:ID" use="required"/>
    <xsd:attribute name="QMZimmer" type="xsd:decimal"
        use="required"/>
</xsd:complexType>

<xsd:complexType name="MieteT">
        <xsd:attribute name="KM" type="xsd:decimal"
            use="_____________________"/>
        <xsd:attribute name="NK" type="xsd:decimal"
            use="_____________________"/>
</xsd:complexType>

</xsd:schema>
```

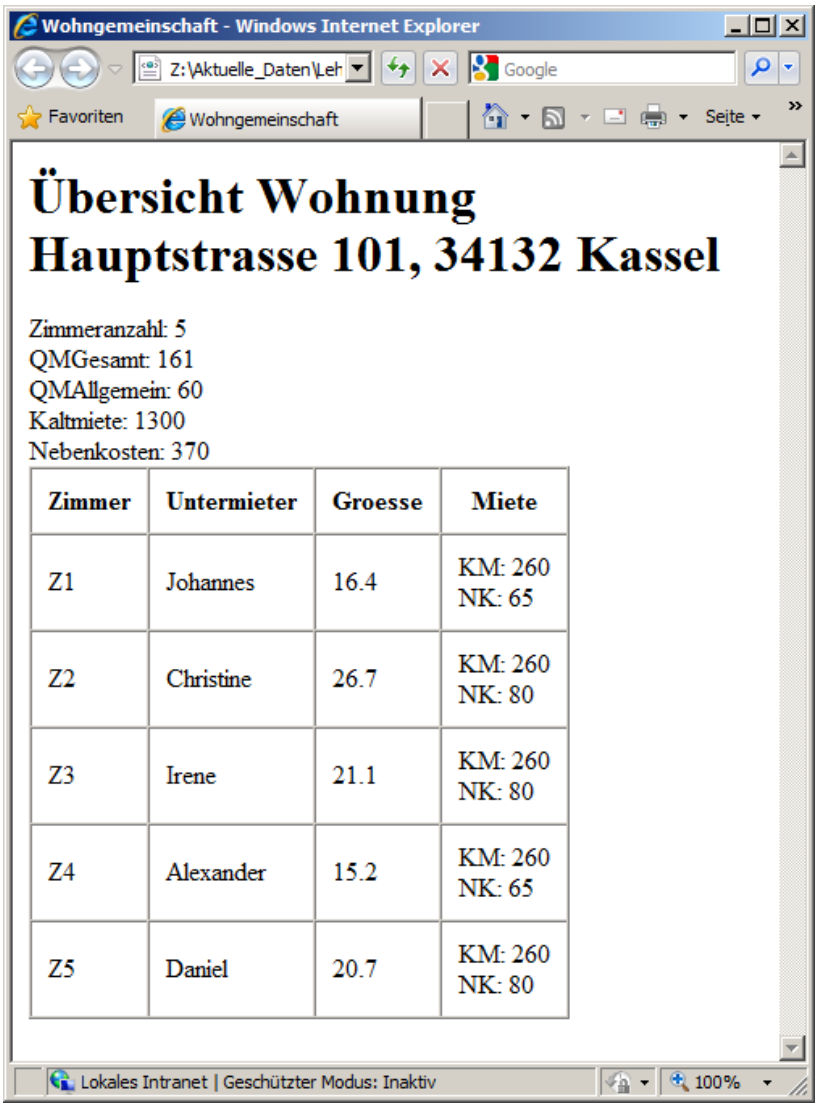
Aufgabe 4:

Ergänzen Sie die Lücken im Stylesheet, damit ein geeignetes Dokument, das der DTD aus Aufgabe 2 genügt, die unten gezeigte Ausgabe erzeugt.

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet version='1.0'
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
    <html>
        <head><title>Wohngemeinschaft</title></head>
        <body>
            <h1>_____ <br/>
                <xsl:value-of select="Wohnung/Adresse"/>
            </h1>
            <xsl:apply-templates/>
        </body>
    </html>
</xsl:template>

<xsl:template match="Wohnung">
    <h2>
        <xsl:for-each select="@*">
            <xsl:value-of select="name()"/>
            <xsl:text>: </xsl:text>
            _____
            <br/>
        </xsl:for-each>
    </h2>
    <table border="1" cellspacing="0" cellpadding="10">
        <tr>
        <th>Zimmer</th><th>Untermieter</th><th>Groesse</th>
                <th>Miete</th>
        </tr>
        <xsl:_____ select="Zimmer"/>
    </table>
</xsl:template>

<xsl:template match="_____">
    <tr>
        <td><xsl:value-of select="@Nummer"/></td>
```

```
        <td><xsl:value-of select="UM"/></td>
        <td><xsl:value-of select="_____________________"/></td>
        <td>
            <xsl:for-each select="_____________________">
                <xsl:value-of select="name()"/>
                <xsl:text>: </xsl:text>
                <xsl:value-of select="."/>
                <br/>
            </xsl:for-each>
        </td>
    </tr>
</xsl:template>
</xsl:stylesheet>
```

Ausgabe:



Aufgabe 5:

Die folgende XQuery soll auf Ihrem Dokument **WG.xml** aus Aufgabe 1 mit den Daten von Aufgabe 4 laufen und Kaltmiete (KM) und Nebenkosten (NK) neu berechnen. Ergänzen Sie die Lücken in der XQuery, damit das Ergebnis unten entsteht!

Hinweis: **fn:round** rundet eine Gleitkommazahl auf eine ganze Zahl. Die Nebenkosten je Zimmer errechnen sich jetzt aus den Nebenkosten der Wohnung geteilt durch die Zimmeranzahl.

```
<Anteilliste>
{
    let $doc := fn:doc("_________________")
    let $anzahl := $doc/Wohnung/@Zimmeranzahl
    let $km := $doc/Wohnung/@Kaltmiete
    let $nk := $doc/Wohnung/@Nebenkosten
    for $zimmer in ____________________________________
    return
     <Mietanteil fuer="_____________________________________">
         <KM>
              { fn:round($km * ($zimmer/@QMZimmer div
                   ($doc/Wohnung/@QMGesamt - $doc/Wohnung/@QMAllgemein))) }
         </KM>
         <NK>
              { fn:round(_____________________________________) }
         </NK>
     </Mietanteil>
}
</Anteilliste>
```

Ergebnis:

```
<Anteilliste>
    <Mietanteil fuer="Johannes">
        <KM>211</KM>
        <NK>74</NK>
    </Mietanteil>
    <Mietanteil fuer="Christine">
        <KM>344</KM>
        <NK>74</NK>
    </Mietanteil>
    <Mietanteil fuer="Irene">
        <KM>272</KM>
        <NK>74</NK>
    </Mietanteil>
    <Mietanteil fuer="Alexander">
        <KM>196</KM>
        <NK>74</NK>
    </Mietanteil>
    <Mietanteil fuer="Daniel">
        <KM>266</KM>
        <NK>74</NK>
    </Mietanteil>
</Anteilliste>
```

Aufgabe 6:

Gegeben sei die folgende Datenbanktabelle mit Namen **WOHNUNGEN**, für die wir durchschnittliche Zimmerpreise ermitteln.

WOHNUNG	QM	ZIMMER	KALTMIETE	NK
Hauptstrasse 101	161	5	1300	370
Platz der Prüfung 3	95	3	820	270
Klausurstrasse 17	120	4	940	310
Diplomallee 4711	155	5	1300	350

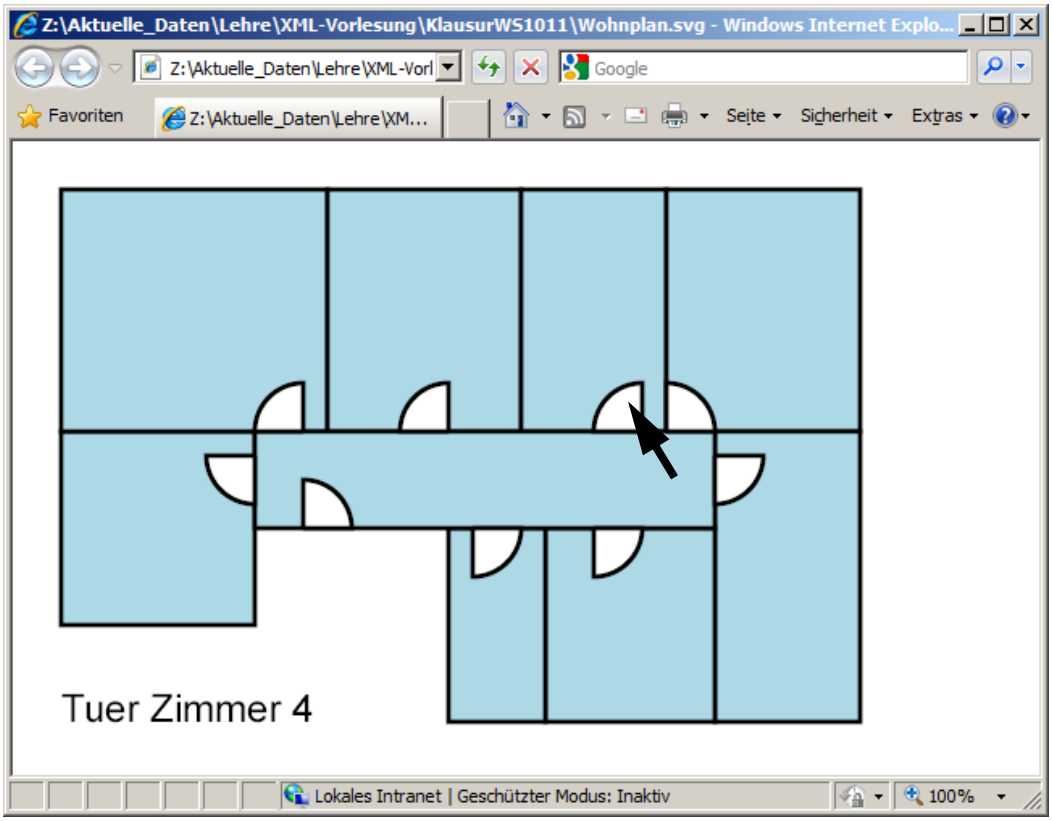
Geben Sie die SQL/XML-Abfrage auf der WOHNUNGEN-Tabelle an, die das Ergebnis unten liefert! Die gezeigten Zahlenwerte ergeben sich aus Kaltmiete + Nebenkosten (NK) geteilt durch die Anzahl der Zimmer. Gesucht sind nur die Angaben für Wohnungen mit 100 oder mehr Quadratmetern.

Ausgabe:

```
<DPreis Adresse="Hauptstrasse 101">334</DPreis>

<DPreis Adresse="Klausurstrasse 17">312,5</DPreis>

<DPreis Adresse="Diplomallee 4711">330</DPreis>
```

Aufgabe 7:

Gezeigt ist ein interaktiver Plan der Wohnung aus Aufgabe 4. Der Text links unten ändert sich je nach dem Element, das mit dem Cursor überstrichen wird. Ergänzen Sie die Lücken!




```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg"
  xmlns:xlink="http://www.w3.org/1999/xlink">
  <rect x="0" y="0" width="1000" height="800" fill="white"/>
  <_________________>
    <path id="____________" fill="white" stroke="black"
      d="M 0,0 L 10,0 A 10 10 0 0 1 0 10 L 0,0"/>
  </___________________>
  <!-- der Script Bereich -->
  <script type="text/javascript">
  <![CDATA[
    function meldung(evt, inORout){
      if (inORout == '________')
      {document.getElementById("____________").firstChild.data
        =evt.target.getAttribute("____________");
      } else {
        document.getElementById("__________").firstChild.data = "-";
    }}
```

```
  ]]>
  </script>
  <text id="meintext" x="10" y="120" font-size="8"
    transform="scale(3)">---</text>
  <g transform="scale(3)"
    style="fill:lightblue; stroke:black; stroke-width:1;"
    onmouseover="meldung(evt, 'in')"
    onmouseout="meldung(evt, 'out')" >

    <rect myid="Zimmer 1" x="10" y="60" width="40" height="40"/>
    <rect myid="Zimmer 2" x="10" y="10" width="55" height="50"/>
    <rect myid="Zimmer 3" x="65" y="10" width="40" height="50"/>
    <rect myid="Zimmer 4" x="105" y="10" width="30" height="50"/>
    <rect myid="Zimmer 5" x="135" y="10" width="40" height="50"/>
    <rect myid="Kueche" x="145" y="60" width="30" height="60"/>
    <rect myid="Bad" x="110" y="80" width="35" height="40"/>
    <rect myid="WC" x="90" y="80" width="20" height="40"/>
    <rect myid="Flur" x="50" y="60" width="95" height="20"/>

    <use xlink:href="#tuer" myid="Tuer Bad"
      transform="translate(120, 80) rotate(0)" />
    <use xlink:href="#tuer" myid="Tuer WC"
      transform="translate(95, 80) rotate(0)" />
    <use xlink:href="#tuer" myid="Tuer Kueche"
      transform="translate(145, 65) rotate(0)" />
    <use xlink:href="#tuer" myid="Tuer Zimmer 5"
      transform="translate(135, 60) rotate(-90)" />
    <use xlink:href="#tuer" myid="Tuer Zimmer 4"
      transform="translate(130, 60) rotate(180)" />
    <use xlink:href="#tuer" myid="Tuer Zimmer 3"
      transform="translate(90, 60) rotate(180)" />
    <use xlink:href="#tuer" myid="Tuer Zimmer 2"
      transform="translate(60, 60) rotate(180)" />
    <use xlink:href="#tuer" myid="Haustuere"
      transform="translate(60, 80) rotate(-90)" />
    <use xlink:href="#tuer" myid="Tuer Zimmer 1"
      transform="translate(50, 65) rotate(90)" />
  </g>
</svg>
```

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

Nachname: Vorname:

Matr.Nr.: Studiengang:

MUSTERLÖSUNG

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 120 Minuten.

Aufgabe	Punkte max.	Punkte erreicht
1	6	
2	2 + 2 + 2 + 2	
3	6	
4	6	
5	6	
6	6	
7	6	
Summe	**44**	

Aufgabe 1:

Die Untermietverhältnisse in einer Wohngemeinschaft (WG) sollen durch ein XML-Dokument **WG.xml** festgehalten werden. Das Format des Dokuments wird in der DTD in Aufgabe 2 definiert. Die Daten für die fünf Zimmer der WG kann man der Stylesheet-Darstellung in Aufgabe 4 entnehmen.

Vervollständigen Sie das Dokument **WG.xml** einschließlich aller Attribute, aber **nur für Wohnung, Adresse und die ersten zwei Zimmer und deren Unterelemente**!

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="WGStyle.xsl" ?>
<!DOCTYPE Wohnung SYSTEM "WG.dtd">
<Wohnung
        Zimmeranzahl="5" QMGesamt="161"
        QMAllgemein="60" Kaltmiete="1300" Nebenkosten="370">
   <Adresse>Hauptstrasse 101, 34132 Kassel</Adresse>
   <Zimmer Nummer="Z1" QMZimmer="16.4">
      <UM>Johannes</UM>
      <Miete KM="260" NK="65"/>
   </Zimmer>
   <Zimmer Nummer="Z2" QMZimmer="26.7">
      <UM>Christine</UM>
      <Miete KM="260" NK="80"/>
   </Zimmer>
   <Zimmer Nummer="Z3" QMZimmer="21.1">
      <UM>Irene</UM>
      <Miete KM="260" NK="80"/>
   </Zimmer>
   <Zimmer Nummer="Z4" QMZimmer="15.2">
      <UM>Alexander</UM>
      <Miete KM="260" NK="65"/>
   </Zimmer>
   <Zimmer Nummer="Z5" QMZimmer="20.7">
      <UM>Daniel</UM>
      <Miete KM="260" NK="80"/>
   </Zimmer>

</Wohnung>
```

zwei Zimmer genügten

Aufgabe 2:

Die DTD für die Wohngemeinschaft aus Aufgabe 1 sieht wie folgt aus.

```
<!-- DTD fuer die Wohngemeinschaft -->
<!ELEMENT Wohnung (Adresse, Zimmer+)>
<!ATTLIST Wohnung    Zimmeranzahl CDATA #REQUIRED
                     QMGesamt CDATA #REQUIRED
                     QMAllgemein CDATA #REQUIRED
                     Kaltmiete CDATA #REQUIRED
                     Nebenkosten CDATA #REQUIRED>

<!ELEMENT Adresse (#PCDATA)>

<!ELEMENT Zimmer (UM, Miete)>
<!ATTLIST Zimmer  Nummer ID #REQUIRED
                  QMZimmer CDATA #REQUIRED>

<!ELEMENT UM (#PCDATA)>

<!ELEMENT Miete EMPTY>
<!ATTLIST Miete  KM CDATA #IMPLIED
                 NK CDATA #IMPLIED>
```

(a) Wie kann man leicht erreichen, dass die DTD für eine **Wohnung** mindestens zwei **Zimmer** verlangt?

<!ELEMENT Wohnung (Adresse, Zimmer, Zimmer+)>

(b) Was bewirkt die Vereinbarung **EMPTY** bei **Miete**?

Das Element „Miete" darf keine Unterelemente und keinen Textinhalt besitzen.

(c) Welche Attribute und Elemente dürfen weggelassen werden?

nur KM und NK in Miete.

(d) Wird verhindert, dass es zwei Zimmer mit gleicher Zimmernummer gibt? Wenn ja, womit erreicht man das; wenn nein, warum leistet diese DTD das nicht?

Ja, erreicht man mit dem Attribut *Nummer* vom Typ ID.

Aufgabe 3:

Ein XML-Schema zum WG-Dokument sieht wie folgt aus.

Füllen Sie die Lücken. Orientieren Sie sich bei den gewünschten Angaben an der DTD aus Aufgabe 2.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<xsd:element name="___Wohnung____" type="___WohnungT___"/>

<xsd:complexType name="WohnungT">
    <xsd:sequence>
        <xsd:element name="Adresse" type="xsd:string"/>
        <xsd:element name="Zimmer" type="ZimmerT"
            minOccurs="1" maxOccurs="10"/>
    </xsd:sequence>
    <xsd:attribute name="Zimmeranzahl" type="xsd:decimal"
            use="__required__"/>
    <xsd:attribute name="QMGesamt" type="xsd:decimal"
            use="__required__"/>
    <xsd:attribute name="QMAllgemein" type="xsd:decimal"
            use="__required__"/>
    <xsd:attribute name="Kaltmiete" type="xsd:decimal"
            use="__required__"/>
    <xsd:attribute name="Nebenkosten" type="xsd:decimal"
            use="__required__"/>

</xsd:complexType>

<xsd:complexType name="ZimmerT">
    <xsd:sequence>
        <xsd:element name="UM" type="xsd:string"/>
        <xsd:element name="Miete" type="___MieteT___"/>
    </xsd:sequence>
    <xsd:attribute name="__Nummer__" type="xsd:ID" use="required"/>
    <xsd:attribute name="QMZimmer" type="xsd:decimal"
        use="required"/>
</xsd:complexType>

<xsd:complexType name="MieteT">
        <xsd:attribute name="KM" type="xsd:decimal"
            use="__optional__"/>
        <xsd:attribute name="NK" type="xsd:decimal"
            use="__optional__"/>
</xsd:complexType>

</xsd:schema>
```

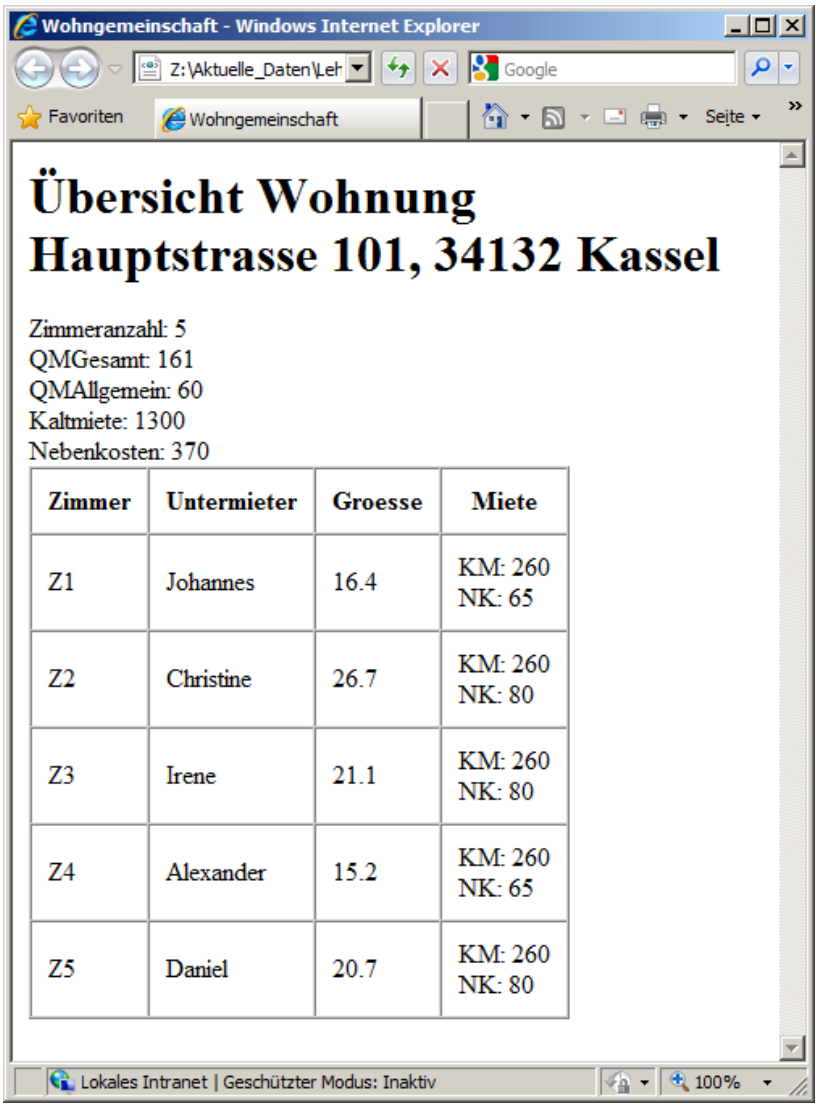
Aufgabe 4:

Ergänzen Sie die Lücken im Stylesheet, damit ein geeignetes Dokument, das der DTD aus Aufgabe 2 genügt, die unten gezeigte Ausgabe erzeugt.

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet version='1.0'
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
    <html>
        <head><title>Wohngemeinschaft</title></head>
        <body>
            <h1>__Übersicht Wohnung__ <br/>
                <xsl:value-of select="Wohnung/Adresse"/>
            </h1>
            <xsl:apply-templates/>
        </body>
    </html>
</xsl:template>

<xsl:template match="Wohnung">
    <h2>
        <xsl:for-each select="@*">
            <xsl:value-of select="name()"/>
            <xsl:text>: </xsl:text>
            __<xsl:value-of select="."/>__
            <br/>
        </xsl:for-each>
    </h2>
    <table border="1" cellspacing="0" cellpadding="10">
        <tr>
        <th>Zimmer</th><th>Untermieter</th><th>Groesse</th>
                <th>Miete</th>
        </tr>
        <xsl:__apply-templates__ select="Zimmer"/>
    </table>
</xsl:template>

<xsl:template match="__Zimmer__">
    <tr>
        <td><xsl:value-of select="@Nummer"/></td>
```

```
        <td><xsl:value-of select="UM"/></td>
        <td><xsl:value-of select="__@QMZimmer__"/></td>
        <td>
            <xsl:for-each select="__Miete/@*__">
                <xsl:value-of select="name()"/>
                <xsl:text>: </xsl:text>
                <xsl:value-of select="."/>
                <br/>
            </xsl:for-each>
        </td>
    </tr>
</xsl:template>
</xsl:stylesheet>
```

Ausgabe:



Aufgabe 5:

Die folgende XQuery soll auf Ihrem Dokument **WG.xml** aus Aufgabe 1 mit den Daten von Aufgabe 4 laufen und Kaltmiete (KM) und Nebenkosten (NK) neu berechnen. Ergänzen Sie die Lücken in der XQuery, damit das Ergebnis unten entsteht!

Hinweis: **fn:round** rundet eine Gleitkommazahl auf eine ganze Zahl. Die Nebenkosten je Zimmer errechnen sich jetzt aus den Nebenkosten der Wohnung geteilt durch die Zimmeranzahl.

```
<Anteilliste>
{
    let $doc := fn:doc("__WG.xml__")
    let $anzahl := $doc/Wohnung/@Zimmeranzahl
    let $km := $doc/Wohnung/@Kaltmiete
    let $nk := $doc/Wohnung/@Nebenkosten
    for $zimmer in $doc/Wohnung/Zimmer
    return
     <Mietanteil fuer="{$zimmer/UM}">
         <KM>
              { fn:round($km * ($zimmer/@QMZimmer div
                   ($doc/Wohnung/@QMGesamt - $doc/Wohnung/@QMAllgemein))) }
         </KM>
         <NK>
              { fn:round($nk div $anzahl) }
         </NK>
     </Mietanteil>
}
</Anteilliste>
```

Ergebnis:

```
<Anteilliste>
    <Mietanteil fuer="Johannes">
        <KM>211</KM>
        <NK>74</NK>
    </Mietanteil>
    <Mietanteil fuer="Christine">
        <KM>344</KM>
        <NK>74</NK>
    </Mietanteil>
    <Mietanteil fuer="Irene">
        <KM>272</KM>
        <NK>74</NK>
    </Mietanteil>
    <Mietanteil fuer="Alexander">
        <KM>196</KM>
        <NK>74</NK>
    </Mietanteil>
    <Mietanteil fuer="Daniel">
        <KM>266</KM>
        <NK>74</NK>
    </Mietanteil>
</Anteilliste>
```

Aufgabe 6:

Gegeben sei die folgende Datenbanktabelle mit Namen **WOHNUNGEN**, für die wir durchschnittliche Zimmerpreise ermitteln.

WOHNUNG	QM	ZIMMER	KALTMIETE	NK
Hauptstrasse 101	161	5	1300	370
Platz der Prüfung 3	95	3	820	270
Klausurstrasse 17	120	4	940	310
Diplomallee 4711	155	5	1300	350

Geben Sie die SQL/XML-Abfrage auf der WOHNUNGEN-Tabelle an, die das Ergebnis unten liefert! Die gezeigten Zahlenwerte ergeben sich aus Kaltmiete + Nebenkosten (NK) geteilt durch die Anzahl der Zimmer. Gesucht sind nur die Angaben für Wohnungen mit 100 oder mehr Quadratmetern.

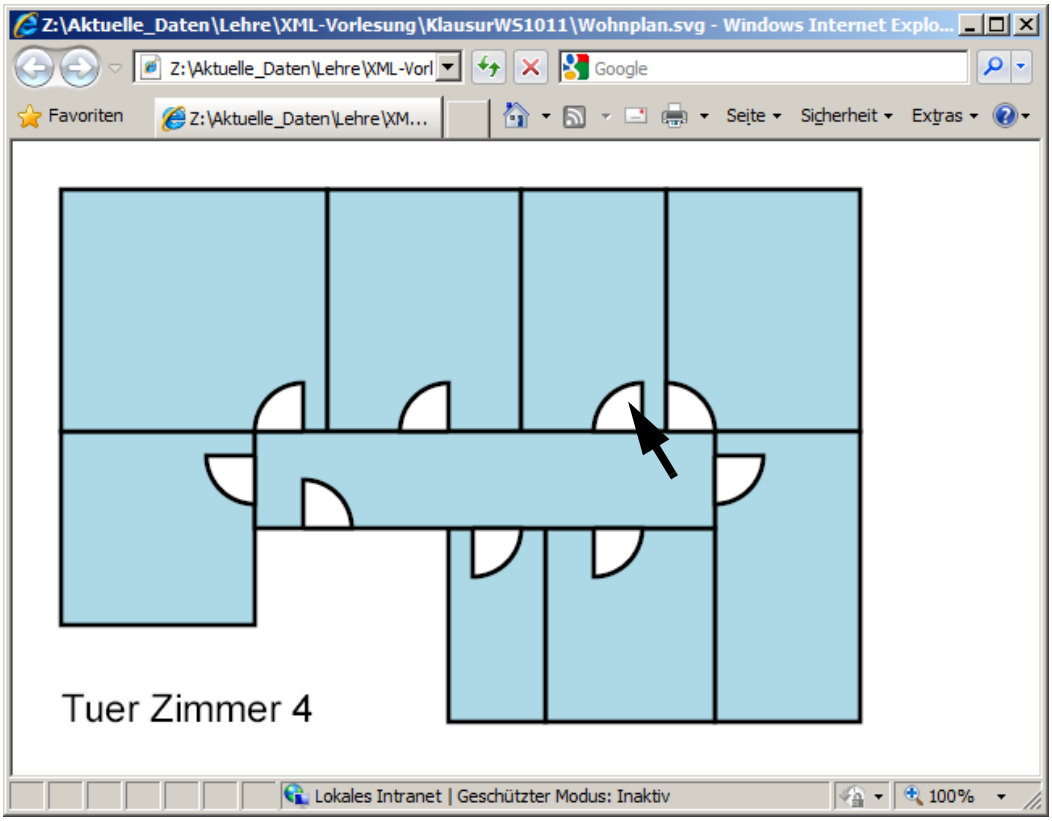
Ausgabe:

```
<DPreis Adresse="Hauptstrasse 101">334</DPreis>

<DPreis Adresse="Klausurstrasse 17">312,5</DPreis>

<DPreis Adresse="Diplomallee 4711">330</DPreis>
```

```
SELECT XMLELEMENT(
        NAME "DPreis",
        XMLATTRIBUTES(WOHNUNG AS "Adresse"),
        (KALTMIETE + NK) / ZIMMER
    )
FROM WOHNUNGEN
WHERE QM >= 100
```

Aufgabe 7:

Gezeigt ist ein interaktiver Plan der Wohnung aus Aufgabe 4. Der Text links unten ändert sich je nach dem Element, das mit dem Cursor überstrichen wird. Ergänzen Sie die Lücken!




```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg"
  xmlns:xlink="http://www.w3.org/1999/xlink">
  <rect x="0" y="0" width="1000" height="800" fill="white"/>
  <__defs__>
    <path id="__tuer__" fill="white" stroke="black"
      d="M 0,0 L 10,0 A 10 10 0 0 1 0 10 L 0,0"/>
  </__defs__>
  <!-- der Script Bereich -->
  <script type="text/javascript">
  <![CDATA[
    function meldung(evt, inORout){
      if (inORout == '__in_')
      {document.getElementById("__meintext__").firstChild.data
        =evt.target.getAttribute("__myid__");
      } else {
        document.getElementById("__meintext__").firstChild.data = "-";
    }}
```

```
  ]]>
  </script>
  <text id="meintext" x="10" y="120" font-size="8"
    transform="scale(3)">---</text>
  <g transform="scale(3)"
    style="fill:lightblue; stroke:black; stroke-width:1;"
    onmouseover="meldung(evt, 'in')"
    onmouseout="meldung(evt, 'out')" >

    <rect myid="Zimmer 1" x="10" y="60" width="40" height="40"/>
    <rect myid="Zimmer 2" x="10" y="10" width="55" height="50"/>
    <rect myid="Zimmer 3" x="65" y="10" width="40" height="50"/>
    <rect myid="Zimmer 4" x="105" y="10" width="30" height="50"/>
    <rect myid="Zimmer 5" x="135" y="10" width="40" height="50"/>
    <rect myid="Kueche" x="145" y="60" width="30" height="60"/>
    <rect myid="Bad" x="110" y="80" width="35" height="40"/>
    <rect myid="WC" x="90" y="80" width="20" height="40"/>
    <rect myid="Flur" x="50" y="60" width="95" height="20"/>

    <use xlink:href="#tuer" myid="Tuer Bad"
      transform="translate(120, 80) rotate(0)" />
    <use xlink:href="#tuer" myid="Tuer WC"
      transform="translate(95, 80) rotate(0)" />
    <use xlink:href="#tuer" myid="Tuer Kueche"
      transform="translate(145, 65) rotate(0)" />
    <use xlink:href="#tuer" myid="Tuer Zimmer 5"
      transform="translate(135, 60) rotate(-90)" />
    <use xlink:href="#tuer" myid="Tuer Zimmer 4"
      transform="translate(130, 60) rotate(180)" />
    <use xlink:href="#tuer" myid="Tuer Zimmer 3"
      transform="translate(90, 60) rotate(180)" />
    <use xlink:href="#tuer" myid="Tuer Zimmer 2"
      transform="translate(60, 60) rotate(180)" />
    <use xlink:href="#tuer" myid="Haustuere"
      transform="translate(60, 80) rotate(-90)" />
    <use xlink:href="#tuer" myid="Tuer Zimmer 1"
      transform="translate(50, 65) rotate(90)" />
  </g>
</svg>
```

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

Nachname: Vorname:

Matr.Nr.: Studiengang:

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 90 Minuten.

Aufgabe	**Punkte max.**	**Punkte erreicht**
1	2	
2	2 + 2 + 1 + 1	
3	6	
4	4	
5	4	
6	5	
7	5	
Summe	**32**	

Aufgabe 1:

Damit man beim Verfassen einer Arbeit nicht den Überblick über die Quellen verliert, entwerfen wir mit **FNoten.xml** eine stark vereinfachte Fußnotenverwaltung als XML-Dokument, das Zitate den Fußnoten zuordnet.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="FNotenStyle.xsl" ?>
<!DOCTYPE Zitate SYSTEM "FNoten.dtd">
<Zitate Arbeit="Dissertation">
    <Zitat FNotenID="F1" Typ="woertlich">
        <Autor>Meier, Gustav</Autor>
        <Autor>Mueller, Gabi</Autor>
        <Titel>Einfuehrung in das richtige Zitieren</Titel>
        <Zeitschrift Band="23" Nr="4" EDatum="Maerz 2011">
            Paedagogischer Rundbrief
        </Zeitschrift>
        <Zusatz Z="S.11f"/>
    </Zitat>
    <Zitat FNotenID="F2" Typ="sinngemaess">
        <Autor>Wagner, Ernst-August Fridolin</Autor>
        <Titel>Wissenschaft als Wunsch und Wille</Titel>
        <BuchBand VerlagsOrt="Muenchen-Berlin-Kassel" Jahr="2007">
            Irrweg-Hansel
        </BuchBand>
        <Zusatz Z="Kap.3"/>
    </Zitat>
</Zitate>
```

Nehmen wir an, ein Übersetzungsprogramm würde alle Attribute des Dokuments oben in **A1**, **A2**, ... umbenennen und die alten Attributnamen – wie am Beispiel unten gezeigt – zu einem Teil der Attributwerte machen.

```
    <Zitat A1="FNotenID:F2" A2="Typ:sinngemaess">
        <Zusatz A1="Z:Kap.3"/>
```

Wäre das entstehende Dokument ein wohlgeformtes XML-Dokument? Kurze Begründung!

Aufgabe 2:

Die DTD für die Zitate aus Aufgabe 1 sieht wie folgt aus.

```
<!-- DTD fuer die Fussnoten -->
<!ELEMENT Zitate (Zitat+)>
<!ATTLIST Zitate Arbeit CDATA #IMPLIED>

<!ELEMENT Zitat (Autor+, Titel, (Zeitschrift | BuchBand), Zusatz )>
<!ATTLIST Zitat  FNotenID ID #REQUIRED
                 Typ (woertlich | sinngemaess | uebersetzt |
                     vergleiche-auch) "woertlich">

<!ELEMENT Autor (#PCDATA)>
<!ELEMENT Titel (#PCDATA)>

<!ELEMENT Zeitschrift (#PCDATA)>
<!ATTLIST Zeitschrift Band CDATA #IMPLIED
                      Nr CDATA #IMPLIED
                      EDatum CDATA #IMPLIED>

<!ELEMENT BuchBand (#PCDATA)>
<!ATTLIST BuchBand   VerlagsOrt CDATA #REQUIRED
                     Jahr CDATA #REQUIRED>

<!ELEMENT Zusatz EMPTY>
<!ATTLIST Zusatz Z CDATA #IMPLIED>
```

(a) Wie könnte man in der DTD erreichen, dass das Element **`Zusatz`** nur nach **`BuchBand`** vorkommt, nicht nach **`Zeitschrift`**?

(b) Welche Aussagen sind richtig? Die Vereinbarung **`EMPTY`** bei **`Zusatz`** bedeutet, dass

( ) keine Unterelemente enthalten sein dürfen.

( ) kein Textinhalt enthalten sein darf.

( ) keine Attribute vorkommen dürfen.

( ) Unterelemente und Textinhalt weggelassen werden **dürfen**, aber nicht **müssen**.

(c) Darf das Attribut **`Typ`** im Element **`Zitat`** im Dokument weggelassen werden?

(d) Wird verhindert, dass zwei Zitate mit gleichem Fußnotenidentifier auftreten können? Wenn ja, womit erreicht man das; wenn nein, warum leistet diese DTD das nicht?

Aufgabe 3:

Ein XML-Schema zum Zitate-Dokument sieht wie folgt aus.

Füllen Sie die Lücken. Orientieren Sie sich bei den gewünschten Angaben an der DTD aus Aufgabe 2.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<xsd:element name="Zitate">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Zitat" type="_________________"
                minOccurs="1" maxOccurs="____________________"/>
        </xsd:sequence>
        <xsd:attribute name="Arbeit" type="____________________"
                use="optional"/>
    </xsd:complexType>
</xsd:element>

<xsd:complexType name="ZitatT">
    <xsd:sequence>
        <xsd:element name="Autor" type="xsd:string"
            minOccurs="1" maxOccurs="unbounded"/>
        <xsd:element name="Titel" type="xsd:string"/>
        <xsd:_________________>
            <xsd:element name="Zeitschrift"
                type="ZeitschriftT"/>
            <xsd:element name="BuchBand"
                type="BuchBandT"/>
        </xsd:_________________>
        <xsd:element name="Zusatz" type="ZusatzT"/>
    </xsd:sequence>
    <xsd:attribute name="Typ" type="TypType" use="optional"
            default="woertlich"/>
    <xsd:attribute name="FNotenID" type="xsd:____________________"
                    use="required"/>
</xsd:complexType>

<xsd:simpleType name="TypType">
    <xsd:restriction ____________________="xsd:string">
        <xsd:enumeration value="woertlich"/>
        <xsd:enumeration value="sinngemaess"/>
        <xsd:enumeration value="uebersetzt"/>
        <xsd:enumeration value="vergleiche-auch"/>
```

```
    </xsd:restriction>
</xsd:simpleType>


<xsd:complexType name="ZusatzT">
        <xsd:attribute name="Z" type="xsd:string"
            use="optional"/>
</xsd:complexType>

<xsd:complexType name="ZeitschriftT">
    <xsd:simpleContent>
        <xsd:_____________________ base="xsd:string">
            <xsd:attribute name="Band" type="xsd:decimal"/>
            <xsd:attribute name="Nr" type="xsd:decimal"/>
            <xsd:attribute name="EDatum" type="xsd:string"/>
        </xsd:_____________________>
    </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="BuchBandT">
    <xsd:simpleContent>
        <xsd:_____________________ base="xsd:string">
            <xsd:attribute name="VerlagsOrt"
                type="xsd:string" use="_____________________"/>
            <xsd:attribute name="Jahr"
                type="xsd:decimal" use="_____________________"/>
        </xsd:_____________________>
    </xsd:simpleContent>
</xsd:complexType>

</xsd:schema>
```

Aufgabe 4:

Ergänzen Sie die Lücken im Stylesheet, damit ein geeignetes Dokument, das der DTD aus Aufgabe 2 genügt, die unten gezeigte Ausgabe erzeugt. Beachten Sie, dass nach einem Attributwert ein Komma ausgegeben werden soll, außer es steht auf der letzten Position.

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet version='1.0'
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
    <html>
    <head><title>Fussnoten</title></head>
        <body>
        <h1>Liste der Fussnoten fuer
            <xsl:value-of select="_____________________________"/>
        </h1>
        _____________________________________________
        </body>
    </html>
</xsl:template>

<xsl:template match="Zitate">
     ____________________________________________________
</xsl:template>

<xsl:template match="Zitat">
    <xsl:value-of select="@FNotenID"/><br/>
    <xsl:value-of select="@Typ"/>
    <xsl:text> zitiert nach:</xsl:text><br/>
    <xsl:for-each select="*">
        <xsl:choose>
        <xsl:when test="normalize-space(text()) != ''">
            <xsl:value-of select="text()"/><br/>
        </xsl:when>
        <xsl:otherwise/>
        </xsl:choose>
        <xsl:for-each select="_______________">
            <xsl:value-of select="."/>
            <xsl:if test="_____________________________ != last()">
                <xsl:text>, </xsl:text>
            </xsl:if>
        </xsl:for-each>
```

```
        <xsl:if test="@*"><br/></xsl:if>
    </xsl:for-each>
    <P/>
</xsl:template>
</xsl:stylesheet>
```

Ausgabe:

Aufgabe 5:

Was liefert die folgende XQuery auf dem Dokument **FNoten.xml** aus Aufgabe 1?

```
<Zeitschrift_woertlich>
{
     let $doc := fn:doc("FNoten.xml")
     for $zitat in $doc/Zitate/Zitat[@Typ = "woertlich" and Zeitschrift]
     return
          $zitat/Titel
}
</Zeitschrift_woertlich>
```

Ergebnis:

```
<Zeitschrift_woertlich>









</Zeitschrift_woertlich>
```

Aufgabe 6:

Gegeben sei die folgende Datenbanktabelle mit Namen **ZITATE**, auf der wir die Anzahl der Autoren je Zitat ausgeben.

FID	AUTOR	QUELLE	JAHR
F1	Meier Gustav	Zeitschrift	2011
F1	Mueller Gabi	Zeitschrift	2011
F2	Wagner Ernst-August Fridolin	Buch	2007
F3	Mueller Gabi	Zeitschrift	2010
F3	Schmidt Egon	Zeitschrift	2010

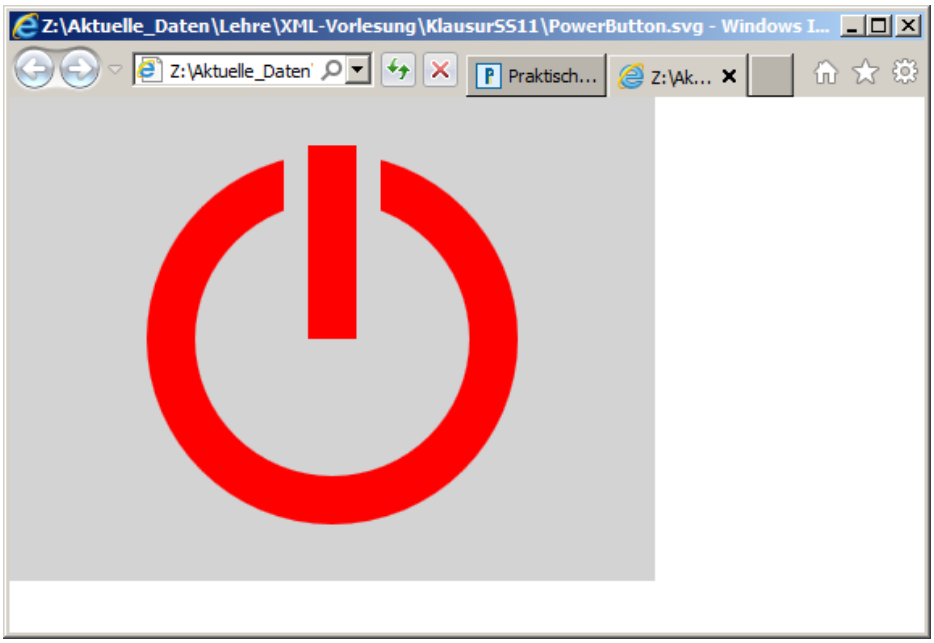
Wie lautet die SQL/XML-Abfrage auf der **ZITATE**-Tabelle, die das gezeigte Ergebnis liefert? Hinweis: Sie benötigen die COUNT-Funktion und GROUP BY.

Ausgabe:

```
<Zitat Autorenanzahl="2">F1</Zitat>
<Zitat Autorenanzahl="1">F2</Zitat>
<Zitat Autorenanzahl="2">F3</Zitat>
```

Aufgabe 7:

Als Ein-/Ausschalter findet man auf vielen Geräten das unten gezeigte Symbol. Ergänzen Sie die fehlenden Stellen, damit das unten gezeigte Bild mittig im gegebenen Rechteck entsteht!




```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg">
  <rect x="0" y="0" width="400" height="300" fill="lightgray"/>
  <circle cx="_____" cy="_____" r="100" fill="none"
    stroke="red" stroke-width="30"/>
  <line x1="_____" y1="20" x2="_____" y2="160" stroke="_______________"
    stroke-width="60"/>
  <line x1="_____" y1="30" x2="_____" y2="____" stroke="red"
    stroke-width="______"/>
</svg>
```

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

MUSTERLÖSUNG

Nachname: Vorname:

Matr.Nr.: Studiengang:

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 90 Minuten.

Aufgabe	Punkte max.	Punkte erreicht
1	2	
2	2 + 2 + 1 + 1	
3	6	
4	4	
5	4	
6	5	
7	5	
Summe	**32**	

Aufgabe 1:

Damit man beim Verfassen einer Arbeit nicht den Überblick über die Quellen verliert, entwerfen wir mit **FNoten.xml** eine stark vereinfachte Fußnotenverwaltung als XML-Dokument, das Zitate den Fußnoten zuordnet.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="FNotenStyle.xsl" ?>
<!DOCTYPE Zitate SYSTEM "FNoten.dtd">
<Zitate Arbeit="Dissertation">
    <Zitat FNotenID="F1" Typ="woertlich">
        <Autor>Meier, Gustav</Autor>
        <Autor>Mueller, Gabi</Autor>
        <Titel>Einfuehrung in das richtige Zitieren</Titel>
        <Zeitschrift Band="23" Nr="4" EDatum="Maerz 2011">
            Paedagogischer Rundbrief
        </Zeitschrift>
        <Zusatz Z="S.11f"/>
    </Zitat>
    <Zitat FNotenID="F2" Typ="sinngemaess">
        <Autor>Wagner, Ernst-August Fridolin</Autor>
        <Titel>Wissenschaft als Wunsch und Wille</Titel>
        <BuchBand VerlagsOrt="Muenchen-Berlin-Kassel" Jahr="2007">
            Irrweg-Hansel
        </BuchBand>
        <Zusatz Z="Kap.3"/>
    </Zitat>
</Zitate>
```

Nehmen wir an, ein Übersetzungsprogramm würde alle Attribute des Dokuments oben in **A1**, **A2**, ... umbenennen und die alten Attributnamen – wie am Beispiel unten gezeigt – zu einem Teil der Attributwerte machen.

```
    <Zitat A1="FNotenID:F2" A2="Typ:sinngemaess">
        <Zusatz A1="Z:Kap.3"/>
```

Wäre das entstehende Dokument ein wohlgeformtes XML-Dokument? Kurze Begründung!

Ja, die Attributnamen innerhalb eines Elements sind eindeutig und der Doppelpunkt ist ein erlaubtes Zeichen in Attributwerten.

Aufgabe 2:

Die DTD für die Zitate aus Aufgabe 1 sieht wie folgt aus.

```
<!-- DTD fuer die Fussnoten -->
<!ELEMENT Zitate (Zitat+)>
<!ATTLIST Zitate Arbeit CDATA #IMPLIED>

<!ELEMENT Zitat (Autor+, Titel, (Zeitschrift | BuchBand), Zusatz )>
<!ATTLIST Zitat  FNotenID ID #REQUIRED
                 Typ (woertlich | sinngemaess | uebersetzt |
                     vergleiche-auch) "woertlich">

<!ELEMENT Autor (#PCDATA)>
<!ELEMENT Titel (#PCDATA)>

<!ELEMENT Zeitschrift (#PCDATA)>
<!ATTLIST Zeitschrift Band CDATA #IMPLIED
                      Nr CDATA #IMPLIED
                      EDatum CDATA #IMPLIED>

<!ELEMENT BuchBand (#PCDATA)>
<!ATTLIST BuchBand   VerlagsOrt CDATA #REQUIRED
                     Jahr CDATA #REQUIRED>

<!ELEMENT Zusatz EMPTY>
<!ATTLIST Zusatz Z CDATA #IMPLIED>
```

(a) Wie könnte man in der DTD erreichen, dass das Element **Zusatz** nur nach **BuchBand** vorkommt, nicht nach **Zeitschrift**?

<!ELEMENT Zitat (Autor+, Titel, (Zeitschrift | (BuchBand, Zusatz)))>

(b) Welche Aussagen sind richtig? Die Vereinbarung **EMPTY** bei **Zusatz** bedeutet, dass

(X) keine Unterelemente enthalten sein dürfen.

(X) kein Textinhalt enthalten sein darf.

( ) keine Attribute vorkommen dürfen.

( ) Unterelemente und Textinhalt weggelassen werden **dürfen**, aber nicht **müssen**.

(c) Darf das Attribut **Typ** im Element **Zitat** im Dokument weggelassen werden?

Ja

(d) Wird verhindert, dass zwei Zitate mit gleichem Fußnotenidentifier auftreten können? Wenn ja, womit erreicht man das; wenn nein, warum leistet diese DTD das nicht?

Ja, weil FNotenID vom Typ ID ist.

Aufgabe 3:

Ein XML-Schema zum Zitate-Dokument sieht wie folgt aus.

Füllen Sie die Lücken. Orientieren Sie sich bei den gewünschten Angaben an der DTD aus Aufgabe 2.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<xsd:element name="Zitate">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Zitat" type="__ZitatT__"
                minOccurs="1" maxOccurs="__unbounded__"/>
        </xsd:sequence>
        <xsd:attribute name="Arbeit" type="__xsd:string__"
                use="optional"/>
    </xsd:complexType>
</xsd:element>

<xsd:complexType name="ZitatT">
    <xsd:sequence>
        <xsd:element name="Autor" type="xsd:string"
            minOccurs="1" maxOccurs="unbounded"/>
        <xsd:element name="Titel" type="xsd:string"/>
        <xsd:__choice__>
            <xsd:element name="Zeitschrift"
                type="ZeitschriftT"/>
            <xsd:element name="BuchBand"
                type="BuchBandT"/>
        </xsd:__choice__>
        <xsd:element name="Zusatz" type="ZusatzT"/>
    </xsd:sequence>
    <xsd:attribute name="Typ" type="TypType" use="optional"
            default="woertlich"/>
    <xsd:attribute name="FNotenID" type="xsd:__ID__"
                    use="required"/>
</xsd:complexType>

<xsd:simpleType name="TypType">
    <xsd:restriction __base__="xsd:string">
        <xsd:enumeration value="woertlich"/>
        <xsd:enumeration value="sinngemaess"/>
        <xsd:enumeration value="uebersetzt"/>
        <xsd:enumeration value="vergleiche-auch"/>
```

```
    </xsd:restriction>
</xsd:simpleType>


<xsd:complexType name="ZusatzT">
        <xsd:attribute name="Z" type="xsd:string"
            use="optional"/>
</xsd:complexType>

<xsd:complexType name="ZeitschriftT">
    <xsd:simpleContent>
        <xsd:__extension__ base="xsd:string">
            <xsd:attribute name="Band" type="xsd:decimal"/>
            <xsd:attribute name="Nr" type="xsd:decimal"/>
            <xsd:attribute name="EDatum" type="xsd:string"/>
        </xsd:__extension__>
    </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="BuchBandT">
    <xsd:simpleContent>
        <xsd:__extension__ base="xsd:string">
            <xsd:attribute name="VerlagsOrt"
                type="xsd:string" use="__required__"/>
            <xsd:attribute name="Jahr"
                type="xsd:decimal" use="__required__"/>
        </xsd:__extension__>
    </xsd:simpleContent>
</xsd:complexType>

</xsd:schema>
```

Aufgabe 4:

Ergänzen Sie die Lücken im Stylesheet, damit ein geeignetes Dokument, das der DTD aus Aufgabe 2 genügt, die unten gezeigte Ausgabe erzeugt. Beachten Sie, dass nach einem Attributwert ein Komma ausgegeben werden soll, außer es steht auf der letzten Position.

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet version='1.0'
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
    <html>
    <head><title>Fussnoten</title></head>
        <body>
        <h1>Liste der Fussnoten fuer
            <xsl:value-of select="__Zitate/@Arbeit__"/>
        </h1>
        __<xsl:apply-templates/>__
        </body>
    </html>
</xsl:template>

<xsl:template match="Zitate">
     __<xsl:apply-templates/>__
</xsl:template>

<xsl:template match="Zitat">
    <xsl:value-of select="@FNotenID"/><br/>
    <xsl:value-of select="@Typ"/>
    <xsl:text> zitiert nach:</xsl:text><br/>
    <xsl:for-each select="*">
        <xsl:choose>
        <xsl:when test="normalize-space(text()) != ''">
            <xsl:value-of select="text()"/><br/>
        </xsl:when>
        <xsl:otherwise/>
        </xsl:choose>
        <xsl:for-each select="__./@*__">          oder auch nur @*
            <xsl:value-of select="."/>
            <xsl:if test="__position()__ != last()">
                <xsl:text>, </xsl:text>
            </xsl:if>
        </xsl:for-each>
```

```
        <xsl:if test="@*"><br/></xsl:if>
    </xsl:for-each>
    <P/>
</xsl:template>
</xsl:stylesheet>
```

Ausgabe:

Aufgabe 5:

Was liefert die folgende XQuery auf dem Dokument **FNoten.xml** aus Aufgabe 1?

```
<Zeitschrift_woertlich>
{
     let $doc := fn:doc("FNoten.xml")
     for $zitat in $doc/Zitate/Zitat[@Typ = "woertlich" and Zeitschrift]
     return
          $zitat/Titel
}
</Zeitschrift_woertlich>
```

Ergebnis:

```
<Zeitschrift_woertlich>

     <Titel>Einfuehrung in das richtige Zitieren</Titel>

</Zeitschrift_woertlich>
```

Aufgabe 6:

Gegeben sei die folgende Datenbanktabelle mit Namen **ZITATE**, auf der wir die Anzahl der Autoren je Zitat ausgeben.

FID	AUTOR	QUELLE	JAHR
F1	Meier Gustav	Zeitschrift	2011
F1	Mueller Gabi	Zeitschrift	2011
F2	Wagner Ernst-August Fridolin	Buch	2007
F3	Mueller Gabi	Zeitschrift	2010
F3	Schmidt Egon	Zeitschrift	2010

Wie lautet die SQL/XML-Abfrage auf der **ZITATE**-Tabelle, die das gezeigte Ergebnis liefert? Hinweis: Sie benötigen die COUNT-Funktion und GROUP BY.

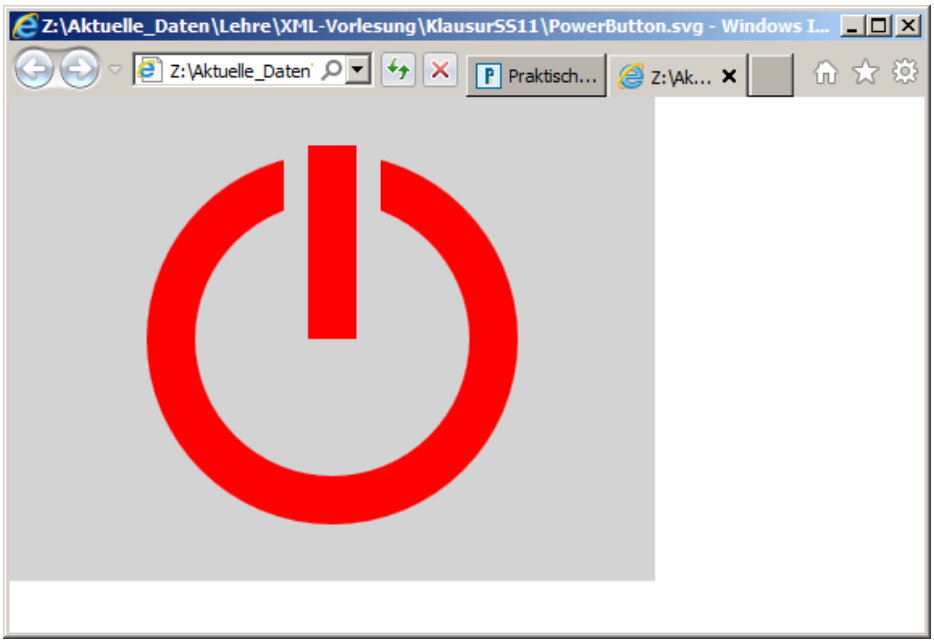
Ausgabe:

```
<Zitat Autorenanzahl="2">F1</Zitat>
<Zitat Autorenanzahl="1">F2</Zitat>
<Zitat Autorenanzahl="2">F3</Zitat>
```

```
SELECT XMLELEMENT(
        NAME "Zitat",
        XMLATTRIBUTES(
          COUNT(*) AS "Autorenanzahl"
        ),
        FID
      )
FROM ZITATE
GROUP BY FID;
```

Aufgabe 7:

Als Ein-/Ausschalter findet man auf vielen Geräten das unten gezeigte Symbol. Ergänzen Sie die fehlenden Stellen, damit das unten gezeigte Bild mittig im gegebenen Rechteck entsteht!




```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg">
  <rect x="0" y="0" width="400" height="300" fill="lightgray"/>
  <circle cx="200" cy="150" r="100" fill="none"
    stroke="red" stroke-width="30"/>
  <line x1="200" y1="20" x2="200" y2="160" stroke="lightgray"
    stroke-width="60"/>
  <line x1="200" y1="30" x2="200" y2="150" stroke="red"
    stroke-width="30"/>
</svg>
```

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

Nachname: Vorname:

Matr.Nr.: Studiengang:

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 120 Minuten.

Aufgabe	**Punkte max.**	**Punkte erreicht**
1	8	
2	3 + 2 + 2	
3	10	
4	7	
5	6	
6	8	
7	6	
Summe	**52**	

In dieser Klausur betrachten wir Stellenangebote einer Universität. Ausgangsdokument für die folgenden Aufgaben ist **`Stellenangebote.xml`** unten.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="AngeboteStyle.xsl" ?>
<!DOCTYPE Stellenangebote SYSTEM "Stellenangebote.dtd">
<Stellenangebote Stand="2012-02-23">
  <Kategorie Art="Professur">
    <Angebot Kennziffer="17363" Besetzung="baldmoeglichst">
      Juniorprofessur fuer Qualitaets- und Prozessmanagement (W1)
      im Fachbereich Maschinenbau
    </Angebot>
    <Angebot Kennziffer="17797" Besetzung="baldmoeglichst">
      W2-Professur Politische Theorie - im FB Gesellschaftswiss.
    </Angebot>
  </Kategorie>
  <Kategorie Art="WissMitarbeiter">
    <Angebot Kennziffer="17737" Besetzung="zum 01.04.2012">
      Wiss. Mitarbeiter/-in (EG 13 TV-H) - im Fachbereich Mathematik
      und Naturwissenschaften - Institut fuer Mathematik
    </Angebot>
    <Angebot Kennziffer="17840">
      Ph.D. position - Analysis of potential CO2 use at the Center
      for Environmental Systems Research, University of Kassel
    </Angebot>
  </Kategorie>
  <Kategorie Art="PaedMitarbeiter">
    <Angebot Kennziffer="17370" Besetzung="spaet. zum 01.08.2012">
      Lehrer/-in als Paed. Mitarbeiter/-in (A13/A14 BBesG)
      im Fachbereich Geistes- und Kulturwissenschaften
    </Angebot>
  </Kategorie>
  <Kategorie Art="Ausbildungsplatz">
    <Angebot Kennziffer="17850">
      Volontaer/-in fuer die Ausbildung zur Redakteurin/Redakteur
    </Angebot>
  </Kategorie>
</Stellenangebote>
```

Aufgabe 1:

Geben Sie eine DTD für die Stellenangebote aus dem Dokument oben an. Es gelten die folgenden Vorgaben:

- Stellenangebote enthält mindestens eine Kategorie.
- Die Angabe des Attributs Stand ist verpflichtend.
- Eine Kategorie enthält kein, ein oder mehrere (unbegrenzt viele) Angebote.
- Das Attribut Art (in Kategorie) ist ein Aufzählungstyp mit mindestens den im Dokument genannten Werten und einem weiteren Wert „andere“, der als Default-Wert dient.
- Das Attribut Kennziffer ist eine ganze Zahl und ist eine Pflichtangabe.
- Das Attribut Besetzung ist optional.

```
<!-- DTD fuer die Stellenangebote-->
```

Aufgabe 2:

(a) Im XML-Dokument oben ist Angebot ein reines Textelement. Üblich ist es, in Stellenangeboten die Besoldung oder Vergütung der Stelle, also W1, W2, A13/A14 BBesG, EG 13 TV-H usw., anzugeben. Wollte man dies als Unterelement im Text mit <BV>...</BV> hervorheben, hätte Angebot **gemischten Inhalt**. Wie lautet die ELEMENT-Vereinbarung für Angebot dann? Beachten Sie, dass BV auch fehlen darf oder mehrfach auftauchen kann wie in

```
... Besoldung nach <BV>A13</BV> oder <BV>A14</BV> je nach ...
```

(b) Kann man in der DTD erzwingen, dass im Zieldokument im Angebot-Element das Attribut Kennziffer vor dem Attribut Besetzung kommt? Wenn ja, wie? Wenn nein, Begründung!

(c) Kann man in der DTD erzwingen, dass die Kategorie-Elemente in einer bestimmten Reihenfolge erscheinen, und zwar geordnet nach dem Attribut Art, also zuerst Professorenstellen, dann Wiss. Mitarbeiterstellen, usw.? Wenn ja, wie? Wenn nein, Begründung!

Aufgabe 3:

Ein XML-Schema zum Stellenangebote-Dokument sieht wie folgt aus. Füllen Sie die Lücken. Orientieren Sie sich bei den gewünschten Angaben an den Vorgaben zur DTD aus Aufgabe 1. Zusätzlich soll gelten:

- Es gibt mindestens eine und maximal zehn Kategorien.
- Das Attribut Stand hat einen geeigneten Datumstyp.
- Das Attribut Kennziffer hat einen ganzzahligen Datentyp.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

```
<xsd:element name="_____________________" type="_________________"/>
<xsd:complexType name="StellenangeboteT">
  <xsd:sequence>
    <xsd:element name="Kategorie" type="KategorieT"
      ____________________________________________________/>
  </xsd:sequence>
  <xsd:attribute name="Stand" type="_____________________"
      use="_________________"/>
</xsd:complexType>
<xsd:complexType name="KategorieT">
  <xsd:sequence>
    <xsd:element name="Angebot" type="AngebotT"
      minOccurs="0" maxOccurs="_____________________"/>
  </xsd:sequence>
  <xsd:attribute name="Art" ___________="andere" use="optional">
    <_____________________________>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="Professur"/>
        <xsd:enumeration value="WissMitarbeiter"/>
        <xsd:enumeration value="PaedMitarbeiter"/>
        <xsd:enumeration value="NichtWissMitarbeiter"/>
        <xsd:enumeration value="Ausbildungsplatz"/>
        <xsd:enumeration value="andere"/>
      </xsd:restriction>
    <_____________________________>
  </xsd:attribute>
</xsd:complexType>
<xsd:complexType name="AngebotT">
  <_____________________________>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="Kennziffer" type="_____________________"
        use="_____________________________"/>
      <xsd:attribute name="Besetzung" type="xsd:string"
        use="optional"/>
    </xsd:extension>
  <_____________________________>
</xsd:complexType>
</xsd:schema>
```

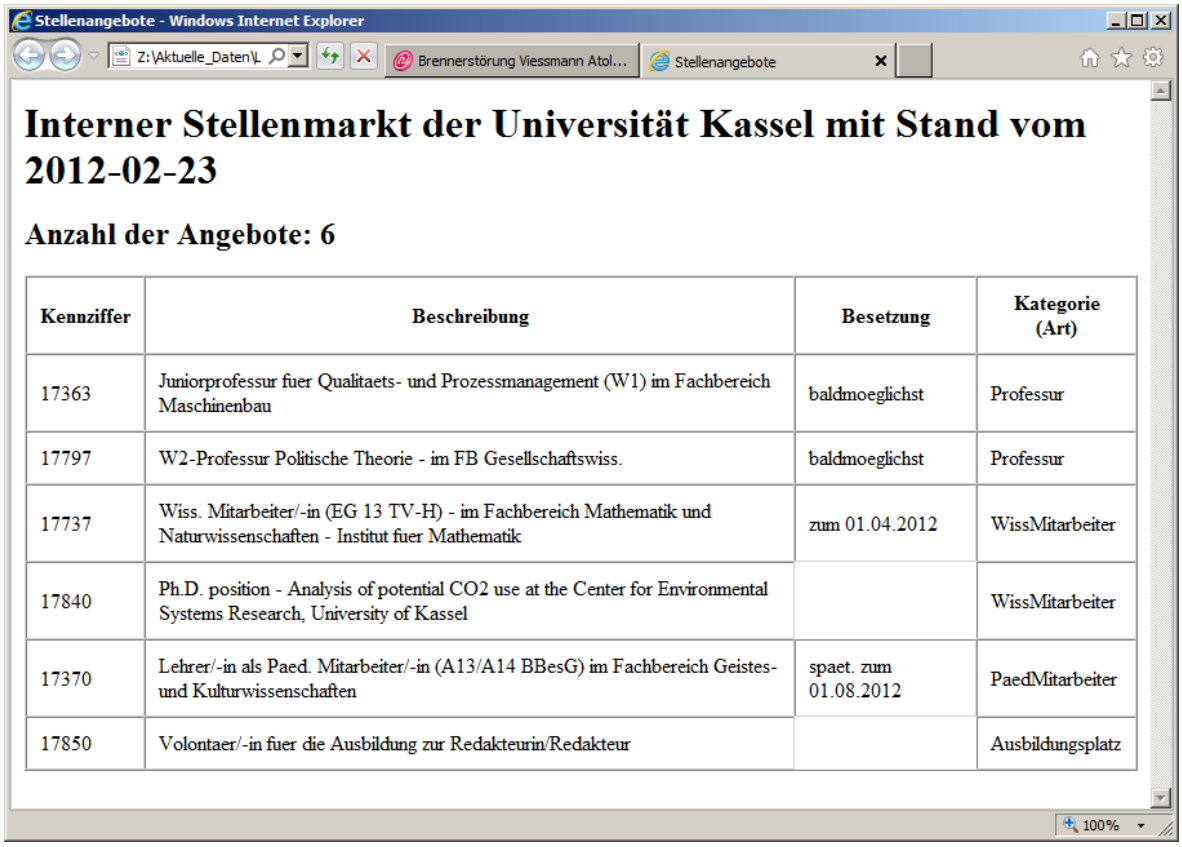
Aufgabe 4:

Ergänzen Sie die Lücken im Stylesheet, damit unser Stellenangebote-Dokument die unten gezeigte Ausgabe erzeugt.

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet version='1.0'
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
    <head><title>Stellenangebote</title></head>
    <body>
      <h1>Interner Stellenmarkt der Universität Kassel mit Stand vom
        <xsl:value-of select="_____________________________________"/>
      </h1>
      <h2>Anzahl der Angebote:
        <xsl:value-of select="_____________________________"/>
      </h2>
      <p/>
      <table border="1" cellspacing="0" cellpadding="10">
        <tr>
          <th>Kennziffer</th><th>Beschreibung</th>
          <th>Besetzung</th><th>Kategorie (Art)</th>
        </tr>
        <xsl:apply-templates select="_____________________"/>
      </table>
    </body>
  </html>
</xsl:template>
<xsl:template match="Angebot">
  <tr>
    <td><xsl:value-of select="_____________________________"/></td>
    <td><xsl:value-of select="_____________________________"/></td>
    <td><xsl:value-of select="_____________________________"/></td>
    <td><xsl:value-of select="_____________________________"/></td>
  </tr>
</xsl:template>
```

```
</xsl:stylesheet>
```

Ausgabe:



Interner Stellenmarkt der Universität Kassel mit Stand vom 2012-02-23

Anzahl der Angebote: 6

Kennziffer	Beschreibung	Besetzung	Kategorie (Art)
17363	Juniorprofessur fuer Qualitaets- und Prozessmanagement (W1) im Fachbereich Maschinenbau	baldmoeglichst	Professur
17797	W2-Professur Politische Theorie - im FB Gesellschaftswiss.	baldmoeglichst	Professur
17737	Wiss. Mitarbeiter/-in (EG 13 TV-H) - im Fachbereich Mathematik und Naturwissenschaften - Institut fuer Mathematik	zum 01.04.2012	WissMitarbeiter
17840	Ph.D. position - Analysis of potential CO2 use at the Center for Environmental Systems Research, University of Kassel		WissMitarbeiter
17370	Lehrer/-in als Paed. Mitarbeiter/-in (A13/A14 BBesG) im Fachbereich Geistes- und Kulturwissenschaften	spaet. zum 01.08.2012	PaedMitarbeiter
17850	Volontaer/-in fuer die Ausbildung zur Redakteurin/Redakteur		Ausbildungsplatz

Aufgabe 5:

Welches Ergebnis liefert die folgende XQuery auf dem Dokument **`Stellenangebote.xml`**.

```
<Angebotssituation Besetzung="baldmoeglichst">
{
     let $doc := fn:doc("Stellenangebote.xml")
     for $kat in $doc//Kategorie
     let $zaehler := fn:count($kat/Angebot[@Besetzung="baldmoeglichst"])
     where $zaehler > 1
     return
       <Sofortangebote Kategorie="{$kat/@Art}">
       {
         $zaehler
       }
       </Sofortangebote>
}
</Angebotssituation>
```

Ergebnis:

Aufgabe 6:

Gegeben sei die folgende Datenbanktabelle mit Namen **STELLENANGEBOTE**.

KATEGORIE	KENN-ZIFFER	BV	FB	BEGINN	ATEXT
Professur	17363	W1	15	baldmoeglichst	Juniorprofessur fuer ...
Professur	17797	W2	5	baldmoeglichst	Professur Politische ...
WissMitarbeiter	17737	EG 13 TV-H	10	zum 01.04.2012	Wiss. Mitarbeiter im FB Mathematik und Naturwiss. ...
PaedMitarbeiter	17370	A13/A14	2	spaetestens zum 01.08.2012	Lehrer/-in als Paed. Mitarbeiter im FB Geistes- ...

Geben Sie die SQL/XML-Abfrage auf der STELLENANGEBOTE-Tabelle an, die als Ergebnis die unten gezeigte Ausgabe ähnlich zu unserem XML-Dokument liefert! Der BV-Wert soll als Attributwert im Element Angebot erscheinen. Hinweis: Sie brauchen ein GROUP BY.

Ausgabe:

```
<Kategorie Art="Professur">
  <Angebot Kennziffer="17363" Besetzung="baldmoeglichst" BV="W1">
    Juniorprofessur fuer ... im Fachbereich Maschinenbau
  </Angebot>
  <Angebot ...
</Kategorie>
<Kategorie ...
```

Aufgabe 7:

Beeinflusst durch ein SVG-Icon für Straßenbahnen auf http://upload.wikimedia.org/wikipedia/commons/c/cf/BSicon_TRAM.svg haben wir ein eigenes für diese Klausur entwickelt, mit dem Ehrgeiz, den Scheibenwischer so zu animieren, dass er unendlich lange wischt. Ergänzen Sie die Lücken und bestimmen Sie die Ersetzungen der Symbole ① bis ④!

Hinweis: Die scale- und translate-Angaben im transform-Attribut bewirken eine Skalierung (Größenänderung) und Koordinatenverschiebung des zugehörigen graphischen Objekts.


```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg"
  xmlns:xlink="http://www.w3.org/1999/xlink">
<rect x="0" y="0" width="600" height="600" fill="white"/>

<def>
  <path id="rumpf"  stroke="black"
    d="M 140,120 L 120,440 A 40 40 0 0 0 160 480 L 440,480
    A 40 40 0 0 0 480 440 L 460 120 A 40 40 0 0 0 420 80
    L 180 80 A 40 40 0 0 0 140 120"/>
  <g id="scheinwerfer">
    <circle _③_="15" _④_="15" r="15" fill="white" stroke="none"/>
    <circle _③_="40" _④_="15" r="15" fill="white" stroke="none"/>
    <rect x="15" y="0" width="25" height="30" stroke="none"
      fill="white"/>
  </g>
</def>

<use xlink:href="_____" fill="black"/>
<use xlink:href="_____" fill="white"
    transform="scale(0.6) translate(200, 100)" />
<use xlink:href="_____" transform="translate(180, 400)" />
<use xlink:href="_____" transform="translate(360, 400)" />
```

```
<line x1="300" y1="80" x2="300" y2="30"
     style="stroke:black; stroke-width:15px; stroke-linecap:round" />
<path style="stroke:black; stroke-width:15px;
     stroke-linecap:round" fill="none"
     d="M 200 45 A 15 15 0 0 1 215 30 L 385 30 A 15 15 0 0 1 400 45" />
<line x1="220" y1="210" x2="220" y2="330" style="stroke:black;
     stroke-width:10px; stroke-linecap:round">
     <animate _①_="x1" values="220; 380; 220" dur="4s"
          repeatCount="_②_"/>
     <animate _①_="y1" values="210; 200; 210; 200; 210"
          dur="4s" repeatCount="_②_"/>
     <animate _①_="x2" values="220; 380; 220" dur="4s"
          repeatCount="_②_"/>
     <animate _①_="y2" values="330; 320; 330; 320; 330"
          dur="4s" repeatCount="_②_"/>
</line>
<line x1="220" y1="280" x2="300" y2="350" style="stroke:black;
     stroke-width:10px; stroke-linecap:round">
     <animate _①_="x1" values="220; 380; 220" dur="4s"
          repeatCount="_②_"/>
     <animate _①_="y1" values="280; 270; 280; 270; 280"
          dur="4s" repeatCount="_②_"/>
</line>

<line x1="200" y1="480" x2="200" y2="500"
     style="stroke:black; stroke-width:20px; stroke-linecap:round" />
<line x1="200" y1="500" x2="160" y2="530"
     style="stroke:black; stroke-width:20px; stroke-linecap:round" />
<line x1="400" y1="480" x2="400" y2="500"
     style="stroke:black; stroke-width:20px; stroke-linecap:round" />
<line x1="400" y1="500" x2="440" y2="530"
     style="stroke:black; stroke-width:20px; stroke-linecap:round" />

</svg>
```

① = _____ ③ = _____

② = _____ ④ = _____

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

Nachname: Vorname:

Matr.Nr.: Studiengang:

MUSTERLÖSUNG

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 120 Minuten.

Aufgabe	Punkte max.	Punkte erreicht
1	8	
2	3 + 2 + 2	
3	10	
4	7	
5	6	
6	8	
7	6	
Summe	**52**	

In dieser Klausur betrachten wir Stellenangebote einer Universität. Ausgangsdokument für die folgenden Aufgaben ist **`Stellenangebote.xml`** unten.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="AngeboteStyle.xsl" ?>
<!DOCTYPE Stellenangebote SYSTEM "Stellenangebote.dtd">
<Stellenangebote Stand="2012-02-23">
  <Kategorie Art="Professur">
    <Angebot Kennziffer="17363" Besetzung="baldmoeglichst">
      Juniorprofessur fuer Qualitaets- und Prozessmanagement (W1)
      im Fachbereich Maschinenbau
    </Angebot>
    <Angebot Kennziffer="17797" Besetzung="baldmoeglichst">
      W2-Professur Politische Theorie - im FB Gesellschaftswiss.
    </Angebot>
  </Kategorie>
  <Kategorie Art="WissMitarbeiter">
    <Angebot Kennziffer="17737" Besetzung="zum 01.04.2012">
      Wiss. Mitarbeiter/-in (EG 13 TV-H) - im Fachbereich Mathematik
      und Naturwissenschaften - Institut fuer Mathematik
    </Angebot>
    <Angebot Kennziffer="17840">
      Ph.D. position - Analysis of potential CO2 use at the Center
      for Environmental Systems Research, University of Kassel
    </Angebot>
  </Kategorie>
  <Kategorie Art="PaedMitarbeiter">
    <Angebot Kennziffer="17370" Besetzung="spaet. zum 01.08.2012">
      Lehrer/-in als Paed. Mitarbeiter/-in (A13/A14 BBesG)
      im Fachbereich Geistes- und Kulturwissenschaften
    </Angebot>
  </Kategorie>
  <Kategorie Art="Ausbildungsplatz">
    <Angebot Kennziffer="17850">
      Volontaer/-in fuer die Ausbildung zur Redakteurin/Redakteur
    </Angebot>
  </Kategorie>
</Stellenangebote>
```

Aufgabe 1:

Geben Sie eine DTD für die Stellenangebote aus dem Dokument oben an. Es gelten die folgenden Vorgaben:

- Stellenangebote enthält mindestens eine Kategorie.
- Die Angabe des Attributs Stand ist verpflichtend.
- Eine Kategorie enthält kein, ein oder mehrere (unbegrenzt viele) Angebote.
- Das Attribut Art (in Kategorie) ist ein Aufzählungstyp mit mindestens den im Dokument genannten Werten und einem weiteren Wert „andere“, der als Default-Wert dient.
- Das Attribut Kennziffer ist eine ganze Zahl und ist eine Pflichtangabe.
- Das Attribut Besetzung ist optional.

```
<!-- DTD fuer die Stellenangebote-->
<!ELEMENT Stellenangebote (Kategorie+)>
<!ATTLIST Stellenangebote Stand CDATA #REQUIRED>

<!ELEMENT Kategorie (Angebot*)>
<!ATTLIST Kategorie Art (Professur | WissMitarbeiter |
                         PaedMitarbeiter | Ausbildungsplatz |
                         andere) "andere">

<!ELEMENT Angebot (#PCDATA)>
<!ATTLIST Angebot Kennziffer CDATA #REQUIRED
                  Besetzung  CDATA #IMPLIED>
```

hier auch NMTOKEN möglich, nicht aber ID

Aufgabe 2:

(a) Im XML-Dokument oben ist Angebot ein reines Textelement. Üblich ist es, in Stellenangeboten die Besoldung oder Vergütung der Stelle, also W1, W2, A13/A14 BBesG, EG 13 TV-H usw., anzugeben. Wollte man dies als Unterelement im Text mit <BV>...</BV> hervorheben, hätte Angebot **gemischten Inhalt**. Wie lautet die ELEMENT-Vereinbarung für Angebot dann? Beachten Sie, dass BV auch fehlen darf oder mehrfach auftauchen kann wie in

```
... Besoldung nach <BV>A13</BV> oder <BV>A14</BV> je nach ...
```

```
<!ELEMENT Angebot (#PCDATA | BV)*>
<!ELEMENT BV (#PCDATA)>
```

(musste nicht angegeben werden)

(b) Kann man in der DTD erzwingen, dass im Zieldokument im Angebot-Element das Attribut Kennziffer vor dem Attribut Besetzung kommt? Wenn ja, wie? Wenn nein, Begründung!

Kann man nicht erzwingen weil Attribute immer ungeordnet sind.

(c) Kann man in der DTD erzwingen, dass die Kategorie-Elemente in einer bestimmten Reihenfolge erscheinen, und zwar geordnet nach dem Attribut Art, also zuerst Professorenstellen, dann Wiss. Mitarbeiterstellen, usw.? Wenn ja, wie? Wenn nein, Begründung!

Elemente sind zwar geordnet, eine Anordnung gleicher Elemente („Sortierung") nach einem Attributwert lässt sich aber in der DTD nicht erzwingen, nur die Anordnung unterschiedlicher Elemente.

Aufgabe 3:

Ein XML-Schema zum Stellenangebote-Dokument sieht wie folgt aus. Füllen Sie die Lücken. Orientieren Sie sich bei den gewünschten Angaben an den Vorgaben zur DTD aus Aufgabe 1. Zusätzlich soll gelten:

- Es gibt mindestens eine und maximal zehn Kategorien.
- Das Attribut Stand hat einen geeigneten Datumstyp.
- Das Attribut Kennziffer hat einen ganzzahligen Datentyp.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

```
<xsd:element name="__Stellenangebote__" type="__StellenangeboteT__"/>
<xsd:complexType name="StellenangeboteT">
  <xsd:sequence>
    <xsd:element name="Kategorie" type="KategorieT"
      __minOccurs="1" maxOccurs="10"__/>
  </xsd:sequence>
  <xsd:attribute name="Stand" type="__xsd:date__"
      use="__required__"/>
</xsd:complexType>
<xsd:complexType name="KategorieT">
  <xsd:sequence>
    <xsd:element name="Angebot" type="AngebotT"
      minOccurs="0" maxOccurs="__unbounded__"/>
  </xsd:sequence>
  <xsd:attribute name="Art" __default__="andere" use="optional">
    <__xsd:simpleType__>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="Professur"/>
        <xsd:enumeration value="WissMitarbeiter"/>
        <xsd:enumeration value="PaedMitarbeiter"/>
        <xsd:enumeration value="NichtWissMitarbeiter"/>
        <xsd:enumeration value="Ausbildungsplatz"/>
        <xsd:enumeration value="andere"/>
      </xsd:restriction>
    <__/xsd:simpleType__>
  </xsd:attribute>
</xsd:complexType>
<xsd:complexType name="AngebotT">
  <__xsd:simpleContent__>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="Kennziffer" type="__xsd:integer__"
        use="__required__"/>
      <xsd:attribute name="Besetzung" type="xsd:string"
        use="optional"/>
    </xsd:extension>
  <__/xsd:simpleContent__>
</xsd:complexType>
</xsd:schema>
```

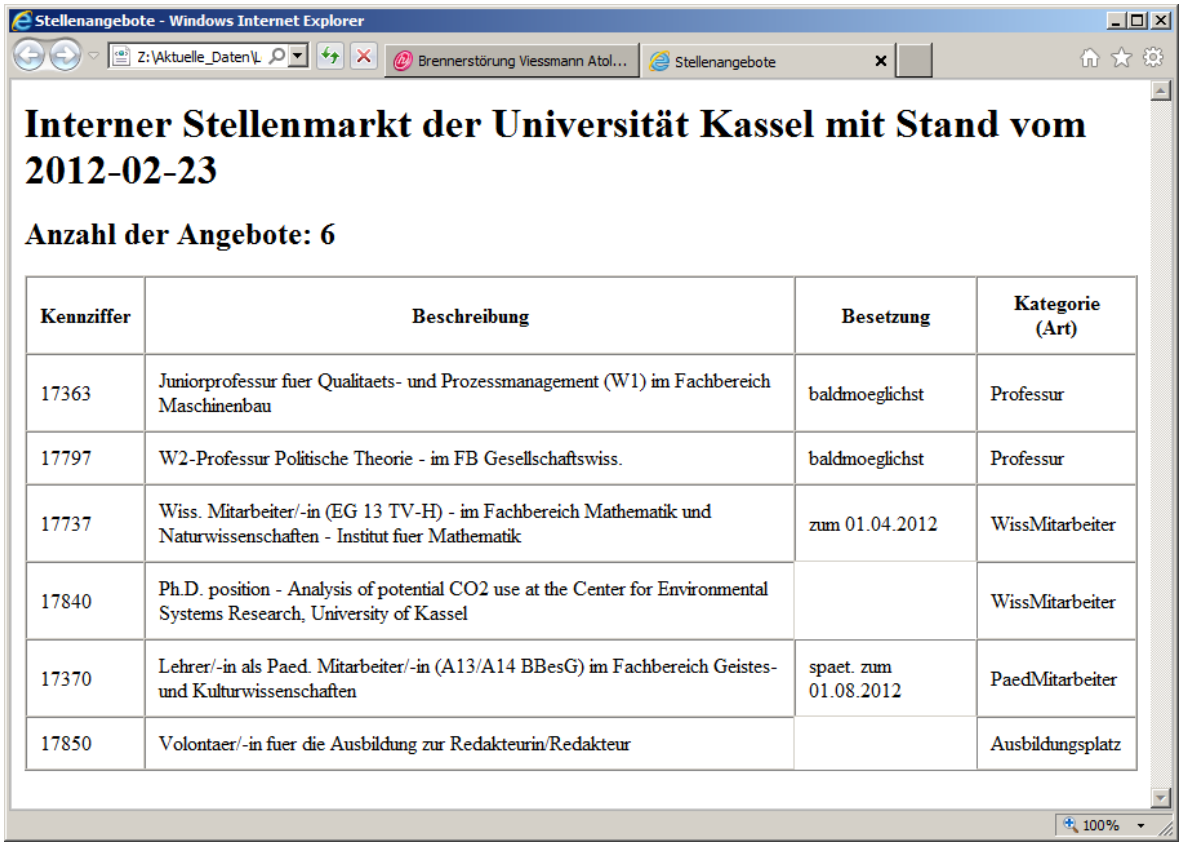
Aufgabe 4:

Ergänzen Sie die Lücken im Stylesheet, damit unser Stellenangebote-Dokument die unten gezeigte Ausgabe erzeugt.

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet version='1.0'
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
    <head><title>Stellenangebote</title></head>
    <body>
      <h1>Interner Stellenmarkt der Universität Kassel mit Stand vom
        <xsl:value-of select="__Stellenangebote/@Stand__"/>
      </h1>
      <h2>Anzahl der Angebote:
        <xsl:value-of select="__count(//Angebot)__"/>
      </h2>
      <p/>
      <table border="1" cellspacing="0" cellpadding="10">
        <tr>
          <th>Kennziffer</th><th>Beschreibung</th>
          <th>Besetzung</th><th>Kategorie (Art)</th>
        </tr>
        <xsl:apply-templates select="__//Angebot__"/>
      </table>
    </body>
  </html>
</xsl:template>
<xsl:template match="Angebot">
  <tr>
    <td><xsl:value-of select="__@Kennziffer__"/></td>
    <td><xsl:value-of select="__.__"/></td>     hier auch text()
    <td><xsl:value-of select="__@Besetzung__"/></td>
    <td><xsl:value-of select="__../@Art__"/></td>
  </tr>
</xsl:template>
```

```
</xsl:stylesheet>
```

Ausgabe:



Interner Stellenmarkt der Universität Kassel mit Stand vom 2012-02-23

Anzahl der Angebote: 6

Kennziffer	Beschreibung	Besetzung	Kategorie (Art)
17363	Juniorprofessur fuer Qualitaets- und Prozessmanagement (W1) im Fachbereich Maschinenbau	baldmoeglichst	Professur
17797	W2-Professur Politische Theorie - im FB Gesellschaftswiss.	baldmoeglichst	Professur
17737	Wiss. Mitarbeiter/-in (EG 13 TV-H) - im Fachbereich Mathematik und Naturwissenschaften - Institut fuer Mathematik	zum 01.04.2012	WissMitarbeiter
17840	Ph.D. position - Analysis of potential CO2 use at the Center for Environmental Systems Research, University of Kassel		WissMitarbeiter
17370	Lehrer/-in als Paed. Mitarbeiter/-in (A13/A14 BBesG) im Fachbereich Geistes- und Kulturwissenschaften	spaet. zum 01.08.2012	PaedMitarbeiter
17850	Volontaer/-in fuer die Ausbildung zur Redakteurin/Redakteur		Ausbildungsplatz

Aufgabe 5:

Welches Ergebnis liefert die folgende XQuery auf dem Dokument **`Stellenangebote.xml`**.

```
<Angebotssituation Besetzung="baldmoeglichst">
{
     let $doc := fn:doc("Stellenangebote.xml")
     for $kat in $doc//Kategorie
     let $zaehler := fn:count($kat/Angebot[@Besetzung="baldmoeglichst"])
     where $zaehler > 1
     return
       <Sofortangebote Kategorie="{$kat/@Art}">
       {
         $zaehler
       }
       </Sofortangebote>
}
</Angebotssituation>
```

Ergebnis:

```
  <Angebotssituation Besetzung="baldmoeglichst">
       <Sofortangebote Kategorie="Professur">2</Sofortangebote>
  </Angebotssituation>
```

Aufgabe 6:

Gegeben sei die folgende Datenbanktabelle mit Namen **STELLENANGEBOTE**.

KATEGORIE	KENN-ZIFFER	BV	FB	BEGINN	ATEXT
Professur	17363	W1	15	baldmoeglichst	Juniorprofessur fuer ...
Professur	17797	W2	5	baldmoeglichst	Professur Politische ...
WissMitarbeiter	17737	EG 13 TV-H	10	zum 01.04.2012	Wiss. Mitarbeiter im FB Mathematik und Naturwiss. ...
PaedMitarbeiter	17370	A13/A14	2	spaetestens zum 01.08.2012	Lehrer/-in als Paed. Mitarbeiter im FB Geistes- ...

Geben Sie die SQL/XML-Abfrage auf der STELLENANGEBOTE-Tabelle an, die als Ergebnis die unten gezeigte Ausgabe ähnlich zu unserem XML-Dokument liefert! Der BV-Wert soll als Attributwert im Element Angebot erscheinen. Hinweis: Sie brauchen ein GROUP BY.

Ausgabe:

```
  <Kategorie Art="Professur">
    <Angebot Kennziffer="17363" Besetzung="baldmoeglichst" BV="W1">
      Juniorprofessur fuer ... im Fachbereich Maschinenbau
    </Angebot>
    <Angebot ...
  </Kategorie>
  <Kategorie ...
```

```
SELECT XMLELEMENT(
  NAME "Kategorie",
    XMLATTRIBUTES(KATEGORIE AS "Art"),
    XMLAGG(
      XMLELEMENT(
        NAME "Angebot",
        XMLATTRIBUTES(KENNZIFFER AS "Kennziffer",
            BEGINN AS "Besetzung",
            BV AS "BV"),
        ATEXT
      )
    )
)
FROM STELLENANGEBOTE
GROUP BY KATEGORIE;
```

Aufgabe 7:

Beeinflusst durch ein SVG-Icon für Straßenbahnen auf http://upload.wikimedia.org/wikipedia/commons/c/cf/BSicon_TRAM.svg haben wir ein eigenes für diese Klausur entwickelt, mit dem Ehrgeiz, den Scheibenwischer so zu animieren, dass er unendlich lange wischt. Ergänzen Sie die Lücken und bestimmen Sie die Ersetzungen der Symbole ① bis ④!

Hinweis: Die scale- und translate-Angaben im transform-Attribut bewirken eine Skalierung (Größenänderung) und Koordinatenverschiebung des zugehörigen graphischen Objekts.


```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg"
  xmlns:xlink="http://www.w3.org/1999/xlink">
<rect x="0" y="0" width="600" height="600" fill="white"/>

<def>
  <path id="rumpf"  stroke="black"
    d="M 140,120 L 120,440 A 40 40 0 0 0 160 480 L 440,480
    A 40 40 0 0 0 480 440 L 460 120 A 40 40 0 0 0 420 80
    L 180 80 A 40 40 0 0 0 140 120"/>
  <g id="scheinwerfer">
    <circle _③_="15" _④_="15" r="15" fill="white" stroke="none"/>
    <circle _③_="40" _④_="15" r="15" fill="white" stroke="none"/>
    <rect x="15" y="0" width="25" height="30" stroke="none"
      fill="white"/>
  </g>
</def>

<use xlink:href="__#rumpf__" fill="black"/>
<use xlink:href="__#rumpf__" fill="white"
    transform="scale(0.6) translate(200, 100)" />
<use xlink:href="__#scheinwerfer__" transform="translate(180, 400)" />
<use xlink:href="__#scheinwerfer__" transform="translate(360, 400)" />
```

```
<line x1="300" y1="80" x2="300" y2="30"
     style="stroke:black; stroke-width:15px; stroke-linecap:round" />
<path style="stroke:black; stroke-width:15px;
     stroke-linecap:round" fill="none"
     d="M 200 45 A 15 15 0 0 1 215 30 L 385 30 A 15 15 0 0 1 400 45" />
<line x1="220" y1="210" x2="220" y2="330" style="stroke:black;
     stroke-width:10px; stroke-linecap:round">
     <animate _①_="x1" values="220; 380; 220" dur="4s"
          repeatCount="_②_"/>
     <animate _①_="y1" values="210; 200; 210; 200; 210"
          dur="4s" repeatCount="_②_"/>
     <animate _①_="x2" values="220; 380; 220" dur="4s"
          repeatCount="_②_"/>
     <animate _①_="y2" values="330; 320; 330; 320; 330"
          dur="4s" repeatCount="_②_"/>
</line>
<line x1="220" y1="280" x2="300" y2="350" style="stroke:black;
     stroke-width:10px; stroke-linecap:round">
     <animate _①_="x1" values="220; 380; 220" dur="4s"
          repeatCount="_②_"/>
     <animate _①_="y1" values="280; 270; 280; 270; 280"
          dur="4s" repeatCount="_②_"/>
</line>

<line x1="200" y1="480" x2="200" y2="500"
     style="stroke:black; stroke-width:20px; stroke-linecap:round" />
<line x1="200" y1="500" x2="160" y2="530"
     style="stroke:black; stroke-width:20px; stroke-linecap:round" />
<line x1="400" y1="480" x2="400" y2="500"
     style="stroke:black; stroke-width:20px; stroke-linecap:round" />
<line x1="400" y1="500" x2="440" y2="530"
     style="stroke:black; stroke-width:20px; stroke-linecap:round" />

</svg>
```

① = __attributeName__

② = __indefinite_____

③ = __cx__

④ = __cy__

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

Nachname: Vorname:

Matr.Nr.: Studiengang:

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 90 Minuten.

Aufgabe	**Punkte max.**	**Punkte erreicht**
1	2 + 1	
2	1 + 1 + 1 + 1	
3	6	
4	6	
5	3 + 2	
6	4	
7	4	
Summe	**32**	

Aufgabe 1:

Wir beschäftigen uns mit Bestsellerlisten von Büchern, wie man sie häufig in Zeitschriften sieht. Die hier gezeigte Datei **`Bestsellerliste.xml`** enthalte eine solche vereinfachte Liste mit drei Büchern.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="BestsellerStyle.xsl" ?>
<Bestseller Genre="Belletristik_Taschenbuch">
    <Buch ISBN="B9783442477890" Preis="9.99">
        <Titel>Das Mädchen auf den Klippen</Titel>
        <Autor>Riley, Lucinda</Autor>
        <Verlag>Goldmann</Verlag>
        <Rang Aktuell="1" Vorwoche="2" Wochen="6"/>
    </Buch>
    <Buch ISBN="B9783492272858" Preis='9.99'>
        <Titel>Das Lächeln der Frauen</Titel>
        <Autor>Barreau, Nicolas</Autor>
        <Autor>Scherrer, Sophie</Autor>
        <Verlag>Piper</Verlag>
        <Rang Vorwoche="1" Aktuell="2" Wochen="14"/>
    </Buch>
    <Buch ISBN="B9783499256356" Preis="8.99">
        <Titel>Tschick</Titel>
        <Autor>Herrndorf, Wolfgang</Autor>
        <Verlag>Rowohlt</Verlag>
        <Rang Wochen="17" Aktuell="3" Vorwoche="5"
            />
    </Buch>
</Bestseller>
```

(a) Ist das Dokument wohlgeformt? Markieren Sie ggf. alle Fehler im Dokument, die das verhindern.

(b) Gibt es ein Element mit gemischtem Inhalt (mixed content). Wenn ja, welches?

Aufgabe 2:

Die DTD für die Bestsellerliste aus Aufgabe 1 (als wohlgeformtes Dokument) sieht wie folgt aus.

```
<!-- DTD fuer die Bestsellerliste -->
<!ELEMENT Bestseller (Buch+)>
<!ATTLIST Bestseller Genre ( Belletristik_Hardcover |
    Belletristik_Taschenbuch | Sachbuch_Hardcover |
    Sachbuch_Taschenbuch ) #REQUIRED>
<!ELEMENT Buch (Titel, Autor+, Verlag, Rang, Zusatz?)>
<!ATTLIST Buch   ISBN ID #REQUIRED
                 Preis CDATA #IMPLIED>
<!ELEMENT Autor (#PCDATA)>
<!ELEMENT Titel (#PCDATA)>
<!ELEMENT Verlag (#PCDATA)>
<!ELEMENT Rang EMPTY>
<!ATTLIST Rang   Aktuell CDATA #REQUIRED
                 Vorwoche CDATA #IMPLIED
                 Wochen CDATA #IMPLIED>
<!ELEMENT Zusatz (#PCDATA)>
```

(a) Die Rangangaben (aktueller Platz, Platz in der Vorwoche, wie lang schon auf der Liste) sind ganze Zahlen. Wie erzwingt man das in der DTD für die Attribute **`Aktuell`**, **`Vorwoche`** und **`Wochen`**?

(b) Was ist richtig? Die Vereinbarung **`EMPTY`** bei **`Rang`** verbietet für dieses Element

( ) Unterelemente.

( ) Textinhalt.

( ) Attribute.

( ) die Schreibweise **`<Rang ...></Rang>`** statt **`<Rang .../>`**.

(c) Darf das Attribut **`Preis`** im Element **`Buch`** im Dokument weggelassen werden?

(d) Hätte man statt **`(Buch+)`** auch **`(Buch, Buch*)`** im Element **`Bestseller`** schreiben dürfen?

Aufgabe 3:

Ein XML-Schema zum Bestseller-Dokument sieht wie folgt aus.

Füllen Sie die Lücken. Orientieren Sie sich bei den gewünschten Angaben an der DTD aus Aufgabe 2. Für die Attribute **`Aktuell`**, **`Vorwoche`**, **`Wochen`** wird der Wertebereich der ganzen Zahlen verlangt.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<xsd:element name="Bestseller">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Buch" type="BuchT"
                minOccurs="___" maxOccurs="_____________________"/>
        </xsd:sequence>
        <xsd:attribute name="_____________" type="_____________"
            use="required"/>
    </xsd:complexType>
</xsd:element>

<xsd:simpleType name="____________________">
    <xsd:_________________ base="xsd:string">
        <xsd:enumeration value="Belletristik_Hardcover"/>
        <xsd:enumeration value="Belletristik_Taschenbuch"/>
        <xsd:enumeration value="Sachbuch_Hardcover"/>
        <xsd:enumeration value="Sachbuch_Taschenbuch"/>
    </xsd:_________________>
</xsd:simpleType>

<xsd:complexType name="BuchT">
    <xsd:sequence>
        <xsd:element name="Titel" type="xsd:string"/>
        <xsd:element name="Autor" type="xsd:string"
            minOccurs="1" maxOccurs="unbounded"/>
        <xsd:element name="Verlag" type="xsd:string"/>
```

```
        <xsd:element name="Rang" type="RangT"/>
        <xsd:element name="Zusatz" type="xsd:string"
            minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="ISBN" type="_____________" use="required"/>
    <xsd:attribute name="Preis" type="xsd:decimal"
        use="_____________________"/>
</xsd:complexType>

<xsd:complexType name="RangT">
    <xsd:attribute name="Aktuell" type="_____________________"
        use="required"/>
    <xsd:attribute name="Vorwoche" type="_____________________"
        use="optional"/>
    <xsd:attribute name="Wochen" type="_____________________"
        use="optional"/>
</xsd:complexType>

</xsd:schema>
```

Aufgabe 4:

Ergänzen Sie die Lücken im Stylesheet, damit ein geeignetes Dokument, das der DTD aus Aufgabe 2 genügt, die unten gezeigte Ausgabe erzeugt.

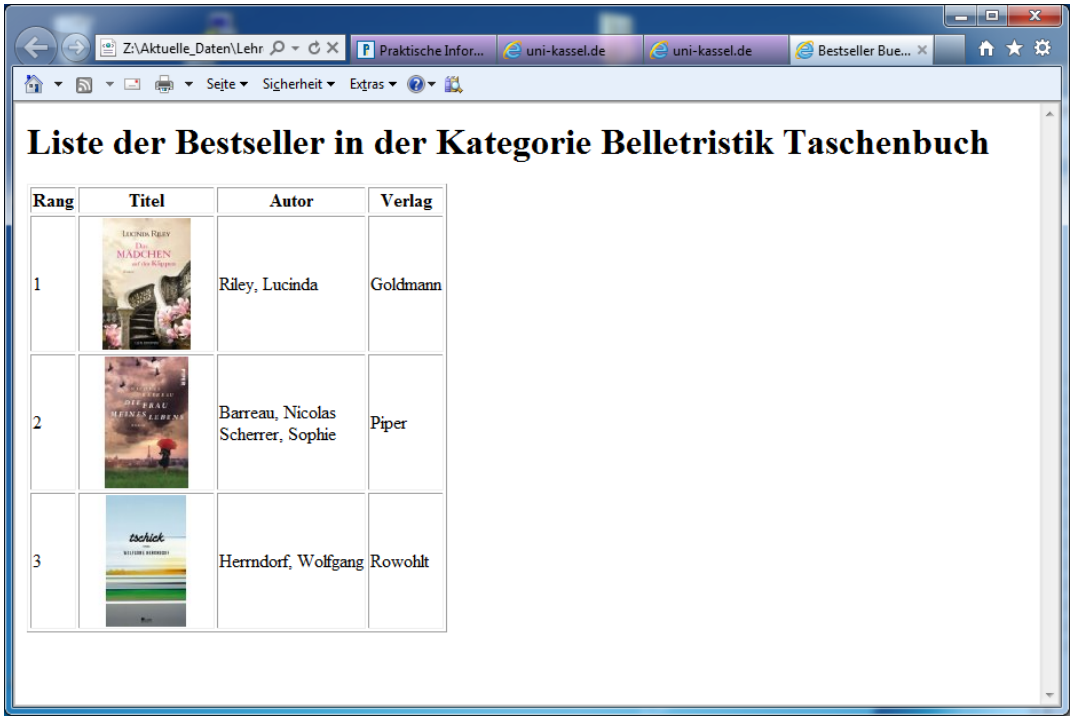
Hinweis: Die Bilder haben als Dateinamen den Wert des ISBN-Attributs, erweitert um die Endung **.jpg**.

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet version='1.0'
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template ______________________________________>
    <html>
        <head><title>Bestseller Buecher</title></head>
        ______________________________________
    </html>
</xsl:template>

<xsl:template match="Bestseller">
    <body>
        <h1>Liste der Bestseller in der Kategorie
            <xsl:value-of select="translate(__________, '_', ' ')"/>
        </h1>
        <_________________ border="1">
            <tr>
                <th>Rang</th><th>Titel</th>
                <th>Autor</th><th>Verlag</th>
            </tr>
            <xsl:apply-templates/>
        <_________________>
    </body>
</xsl:template>

<xsl:template match="Buch">
    <tr>
        <td><xsl:value-of select="_____________________________"/></td>
        <td><img src="{@ISBN}.jpg"/></td>
        <td><_____________________________ select="Autor"/></td>
        <td><_____________________________ select="Verlag"/></td>
    </tr>
</xsl:template>
```

Dies ist ein Unterstrich, keine Lücke!

```
<xsl:template match="Autor">
    <xsl:value-of select="_____________"/><br/>
</xsl:template>

</xsl:stylesheet>
```

Ausgabe:



Liste der Bestseller in der Kategorie Belletristik Taschenbuch

Rang	Titel	Autor	Verlag
1		Riley, Lucinda	Goldmann
2		Barreau, Nicolas Scherrer, Sophie	Piper
3		Herrndorf, Wolfgang	Rowohlt

Aufgabe 5:

Ergänzen Sie die folgende XQuery für Bestseller-Dokumente wie **`Bestsellerliste.xml`** aus Aufgabe 1 (als wohlgeformtes Dokument). Die Abfrage soll den Buchtitel genau der Bücher liefern, die mehr als einen Autor haben. Geben Sie dann auch das Ergebnis für **`Bestsellerliste.xml`** aus Aufgabe 1 an.

```
<Mehrautorenliste>
{
    let $doc := fn:doc("Bestsellerliste.xml")
    for $titel in $doc/Bestseller/Buch/Titel
    where ____________________________________________
    return
        ____________________________
}
</Mehrautorenliste>
```

Ausgabe:

Aufgabe 6:

Gegeben sei die folgende Datenbanktabelle mit Namen **VERLAGSRENNER**.

VERLAG	TITEL	RANG	VORWOCHE
Fischer	Die hellen Tage	1	5
Piper	Das Lächeln der Frauen	2	1
Goldmann	Das Mädchen auf den Klippen	3	2
Diana	Die fernen Stunden	4	6
Rowohlt	Tschick	5	3

Wir suchen die Titel der Bücher, die aufgestiegen sind, d. h. deren Rang kleiner ist als der Vorwochenwert. Wie lautet die zugehörige SQL/XML-Abfrage auf der **VERLAGSRENNER**-Tabelle? Die Ausgabe soll das unten gezeigte Format haben.

```
<Titel Rang="1" Vorwoche="5">Die hellen Tage</Titel>
<Titel Rang="4" Vorwoche="6">Die fernen Stunden</Titel>
```

Aufgabe 7:

In unserer Sammlung animierter Verkehrszeichen betrachten wir heute das *Gefahrzeichen 133 Fußgänger*. In unserer leicht modifizierten Darstellung sind die „Beine“ animiert, d. h. der Fußgänger läuft auf der Stelle. Ergänzen Sie den fehlenden Teil für das rechte Bein!


```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg"
  xmlns:xlink="http://www.w3.org/1999/xlink">
<rect x="0" y="0" width="600" height="600" fill="white"/>

<line x1="50" y1="500" x2="550" y2="500"
  style="stroke:red; stroke-width:50px; stroke-linecap:round"/>
<line x1="50" y1="500" x2="300" y2="67"
  style="stroke:red; stroke-width:50px; stroke-linecap:round"/>
<line x1="300" y1="67" x2="550" y2="500"
  style="stroke:red; stroke-width:50px; stroke-linecap:round"/>
<path style="stroke:black; stroke-width:2px" fill="none"
  d="M 260 50 A 43 43 60 0 1 340 50 L 590 480
    A 43 43 -60 0 1 550 545 L 50 545
    A 43 43 -60 0 1 10 480 L 260 50" />
```

```
<circle id="head" cx="298" cy="224" r="20" fill="black"
stroke="none"/>
<line id="rump" x1="310" y1="270" x2="310" y2="335"
  style="stroke:black; stroke-width:40px; stroke-linecap:round"/>
<line id="leftleg" x1="310" y1="340" x2="250" y2="440"
  style="stroke:black; stroke-width:20px; stroke-linecap:round">
  <animate attributeName="x2" values="250; 370; 250" dur="4s"
    repeatCount="indefinite"/>
</line>

<line id="rightleg"









</line>

<line x1="280" y1="345" x2="340" y2="345"
  style="stroke:white; stroke-width:10px"/>
<line id="leftarm" x1="254" y1="335" x2="306" y2="260"
  style="stroke:black; stroke-width:20px; stroke-linecap:round"/>
<line id="rightarm1" x1="316" y1="260" x2="356" y2="290"
  style="stroke:black; stroke-width:20px; stroke-linecap:round"/>
<line id="rightarm2" x1="356" y1="290" x2="356" y2="330"
  style="stroke:black; stroke-width:20px; stroke-linecap:round"/>
</svg>
```

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

Nachname: **Vorname:**

Matr.Nr.: **Studiengang:**

MUSTERLÖSUNG

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 90 Minuten.

Aufgabe	Punkte max.	Punkte erreicht
1	2 + 1	
2	1 + 1 + 1 + 1	
3	6	
4	6	
5	3 + 2	
6	4	
7	4	
Summe	**32**	

Aufgabe 1:

Wir beschäftigen uns mit Bestsellerlisten von Büchern, wie man sie häufig in Zeitschriften sieht. Die hier gezeigte Datei **`Bestsellerliste.xml`** enthalte eine solche vereinfachte Liste mit drei Büchern.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="BestsellerStyle.xsl" ?>
<Bestseller Genre="Belletristik_Taschenbuch">
    <Buch ISBN="B9783442477890" Preis="9.99">
        <Titel>Das Mädchen auf den Klippen</Titel>
        <Autor>Riley, Lucinda</Autor>
        <Verlag>Goldmann</Verlag>
        <Rang Aktuell="1" Vorwoche="2" Wochen="6"/>
    </Buch>
    <Buch ISBN="B9783492272858" Preis='9.99'>
        <Titel>Das Lächeln der Frauen</Titel>
        <Autor>Barreau, Nicolas</Autor>
        <Autor>Scherrer, Sophie</Autor>
        <Verlag>Piper</Verlag>
        <Rang Vorwoche="1" Aktuell="2" Wochen="14"/>
    </Buch>
    <Buch ISBN="B9783499256356" Preis="8.99">
        <Titel>Tschick</Titel>
        <Autor>Herrndorf, Wolfgang</Autor>
        <Verlag>Rowohlt</Verlag>
        <Rang Wochen="17" Aktuell="3" Vorwoche="5"
            />
    </Buch>
</Bestseller>
```

(a) Ist das Dokument wohlgeformt? Markieren Sie ggf. alle Fehler im Dokument, die das verhindern.

Ja!

(b) Gibt es ein Element mit gemischtem Inhalt (mixed content). Wenn ja, welches?

Nein!

Aufgabe 2:

Die DTD für die Bestsellerliste aus Aufgabe 1 (als wohlgeformtes Dokument) sieht wie folgt aus.

```
<!-- DTD fuer die Bestsellerliste -->
<!ELEMENT Bestseller (Buch+)>
<!ATTLIST Bestseller Genre ( Belletristik_Hardcover |
    Belletristik_Taschenbuch | Sachbuch_Hardcover |
    Sachbuch_Taschenbuch ) #REQUIRED>
<!ELEMENT Buch (Titel, Autor+, Verlag, Rang, Zusatz?)>
<!ATTLIST Buch   ISBN ID #REQUIRED
                 Preis CDATA #IMPLIED>
<!ELEMENT Autor (#PCDATA)>
<!ELEMENT Titel (#PCDATA)>
<!ELEMENT Verlag (#PCDATA)>
<!ELEMENT Rang EMPTY>
<!ATTLIST Rang   Aktuell CDATA #REQUIRED
                 Vorwoche CDATA #IMPLIED
                 Wochen CDATA #IMPLIED>
<!ELEMENT Zusatz (#PCDATA)>
```

(a) Die Rangangaben (aktueller Platz, Platz in der Vorwoche, wie lang schon auf der Liste) sind ganze Zahlen. Wie erzwingt man das in der DTD für die Attribute **Aktuell**, **Vorwoche** und **Wochen**?

Ist in einer DTD nicht möglich.

(b) Was ist richtig? Die Vereinbarung **EMPTY** bei **Rang** verbietet für dieses Element

(X) Unterelemente.

(X) Textinhalt.

( ) Attribute.

( ) die Schreibweise **<Rang ...></Rang>** statt **<Rang .../>**.

(c) Darf das Attribut **Preis** im Element **Buch** im Dokument weggelassen werden?

Ja!

(d) Hätte man statt **(Buch+)** auch **(Buch, Buch*)** im Element **Bestseller** schreiben dürfen?

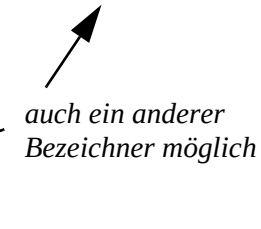
Ja!

Aufgabe 3:

Ein XML-Schema zum Bestseller-Dokument sieht wie folgt aus.

Füllen Sie die Lücken. Orientieren Sie sich bei den gewünschten Angaben an der DTD aus Aufgabe 2. Für die Attribute **`Aktuell`**, **`Vorwoche`**, **`Wochen`** wird der Wertebereich der ganzen Zahlen verlangt.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<xsd:element name="Bestseller">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Buch" type="BuchT"
                minOccurs="_1_" maxOccurs="_unbounded_"/>
        </xsd:sequence>
        <xsd:attribute name="__Genre__" type="__GenreT__"
            use="required"/>
    </xsd:complexType>
</xsd:element>

<xsd:simpleType name="__GenreT__">
    <xsd:__restriction__ base="xsd:string">
        <xsd:enumeration value="Belletristik_Hardcover"/>
        <xsd:enumeration value="Belletristik_Taschenbuch"/>
        <xsd:enumeration value="Sachbuch_Hardcover"/>
        <xsd:enumeration value="Sachbuch_Taschenbuch"/>
    </xsd:__restriction__>
</xsd:simpleType>

<xsd:complexType name="BuchT">
    <xsd:sequence>
        <xsd:element name="Titel" type="xsd:string"/>
        <xsd:element name="Autor" type="xsd:string"
            minOccurs="1" maxOccurs="unbounded"/>
        <xsd:element name="Verlag" type="xsd:string"/>
```



```
        <xsd:element name="Rang" type="RangT"/>
        <xsd:element name="Zusatz" type="xsd:string"
            minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="ISBN" type="__xsd:ID__" use="required"/>
    <xsd:attribute name="Preis" type="xsd:decimal"
        use="__optional__"/>
</xsd:complexType>

<xsd:complexType name="RangT">
    <xsd:attribute name="Aktuell" type="__xsd:integer__"
        use="required"/>
    <xsd:attribute name="Vorwoche" type="__xsd:integer__"
        use="optional"/>
    <xsd:attribute name="Wochen" type="__xsd:integer__"
        use="optional"/>
</xsd:complexType>

</xsd:schema>
```

Aufgabe 4:

Ergänzen Sie die Lücken im Stylesheet, damit ein geeignetes Dokument, das der DTD aus Aufgabe 2 genügt, die unten gezeigte Ausgabe erzeugt.

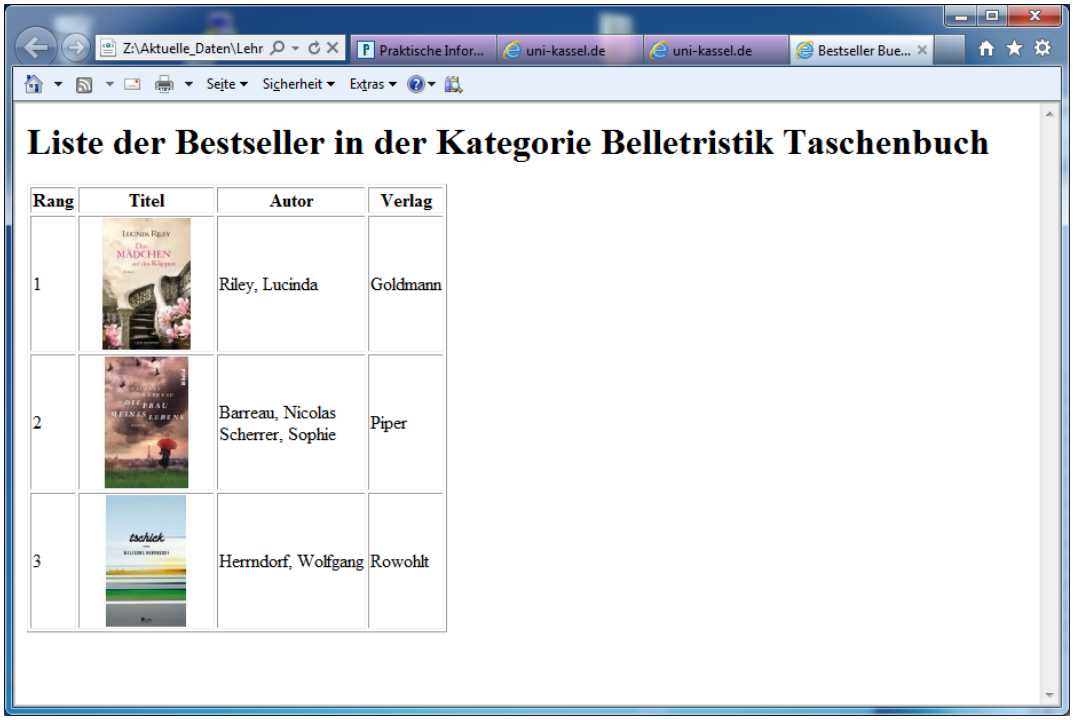
Hinweis: Die Bilder haben als Dateinamen den Wert des ISBN-Attributs, erweitert um die Endung **.jpg**.

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet version='1.0'
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template __match="/"__>
    <html>
        <head><title>Bestseller Buecher</title></head>
        __<xsl:apply-templates/>__
    </html>
</xsl:template>

<xsl:template match="Bestseller">
    <body>
        <h1>Liste der Bestseller in der Kategorie
            <xsl:value-of select="translate(_@Genre_, '_', ' ')"/>
        </h1>
        <__table_ border="1">
            <tr>
                <th>Rang</th><th>Titel</th>
                <th>Autor</th><th>Verlag</th>
            </tr>
            <xsl:apply-templates/>
        <_/table_>
    </body>
</xsl:template>

<xsl:template match="Buch">
    <tr>
        <td><xsl:value-of select="__Rang/@Aktuell__"/></td>
        <td><img src="{@ISBN}.jpg"/></td>
        <td><__xsl:apply-templates__ select="Autor"/></td>
        <td><__xsl:value-of__ select="Verlag"/></td>
    </tr>
</xsl:template>
```

Dies ist ein Unterstrich, keine Lücke!

```
<xsl:template match="Autor">
    <xsl:value-of select="__.__"/><br/>
</xsl:template>

</xsl:stylesheet>
```

Ausgabe:



Liste der Bestseller in der Kategorie Belletristik Taschenbuch

Rang	Titel	Autor	Verlag
1		Riley, Lucinda	Goldmann
2		Barreau, Nicolas Scherrer, Sophie	Piper
3		Herrndorf, Wolfgang	Rowohlt

Aufgabe 5:

Ergänzen Sie die folgende XQuery für Bestseller-Dokumente wie **`Bestsellerliste.xml`** aus Aufgabe 1 (als wohlgeformtes Dokument). Die Abfrage soll den Buchtitel genau der Bücher liefern, die mehr als einen Autor haben. Geben Sie dann auch das Ergebnis für **`Bestsellerliste.xml`** aus Aufgabe 1 an.

```
<Mehrautorenliste>
{
     let $doc := fn:doc("Bestsellerliste.xml")
     for $titel in $doc/Bestseller/Buch/Titel
     where ___count($titel/../Autor) > 1___
     return
          ___$titel___
}
</Mehrautorenliste>
```

Ausgabe:

```
<Mehrautorenliste>
     <Titel>Das Lächeln der Frauen</Titel>
</Mehrautorenliste>
```

Aufgabe 6:

Gegeben sei die folgende Datenbanktabelle mit Namen **VERLAGSRENNER**.

VERLAG	TITEL	RANG	VORWOCHE
Fischer	Die hellen Tage	1	5
Piper	Das Lächeln der Frauen	2	1
Goldmann	Das Mädchen auf den Klippen	3	2
Diana	Die fernen Stunden	4	6
Rowohlt	Tschick	5	3

Wir suchen die Titel der Bücher, die aufgestiegen sind, d. h. deren Rang kleiner ist als der Vorwochenwert. Wie lautet die zugehörige SQL/XML-Abfrage auf der **VERLAGSRENNER**-Tabelle? Die Ausgabe soll das unten gezeigte Format haben.

```
<Titel Rang="1" Vorwoche="5">Die hellen Tage</Titel>
<Titel Rang="4" Vorwoche="6">Die fernen Stunden</Titel>
```

```
SELECT
XMLELEMENT(
    NAME "Titel",
    XMLATTRIBUTES(
        RANG AS "Rang",
        VORWOCHE AS "Vorwoche"
    ),
    TITEL
)
FROM VERLAGSRENNER
WHERE RANG < VORWOCHE
```

Aufgabe 7:

In unserer Sammlung animierter Verkehrszeichen betrachten wir heute das *Gefahrzeichen 133 Fußgänger*. In unserer leicht modifizierten Darstellung sind die „Beine“ animiert, d. h. der Fußgänger läuft auf der Stelle. Ergänzen Sie den fehlenden Teil für das rechte Bein!


```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg"
  xmlns:xlink="http://www.w3.org/1999/xlink">
<rect x="0" y="0" width="600" height="600" fill="white"/>

<line x1="50" y1="500" x2="550" y2="500"
  style="stroke:red; stroke-width:50px; stroke-linecap:round"/>
<line x1="50" y1="500" x2="300" y2="67"
  style="stroke:red; stroke-width:50px; stroke-linecap:round"/>
<line x1="300" y1="67" x2="550" y2="500"
  style="stroke:red; stroke-width:50px; stroke-linecap:round"/>
<path style="stroke:black; stroke-width:2px" fill="none"
  d="M 260 50 A 43 43 60 0 1 340 50 L 590 480
    A 43 43 -60 0 1 550 545 L 50 545
    A 43 43 -60 0 1 10 480 L 260 50" />
```

```
<circle id="head" cx="298" cy="224" r="20" fill="black"
stroke="none"/>
<line id="rump" x1="310" y1="270" x2="310" y2="335"
  style="stroke:black; stroke-width:40px; stroke-linecap:round"/>
<line id="leftleg" x1="310" y1="340" x2="250" y2="440"
  style="stroke:black; stroke-width:20px; stroke-linecap:round">
  <animate attributeName="x2" values="250; 370; 250" dur="4s"
    repeatCount="indefinite"/>
</line>

<line id="rightleg" x1="310" y1="340" x2="370" y2="440"
  style="stroke:black;
  stroke-width:20px;
  stroke-linecap:round">
  <animate attributeName="x2" values="370; 250; 370" dur="4s"
    repeatCount="indefinite"/>
</line>

<line x1="280" y1="345" x2="340" y2="345"
  style="stroke:white; stroke-width:10px"/>
<line id="leftarm" x1="254" y1="335" x2="306" y2="260"
  style="stroke:black; stroke-width:20px; stroke-linecap:round"/>
<line id="rightarm1" x1="316" y1="260" x2="356" y2="290"
  style="stroke:black; stroke-width:20px; stroke-linecap:round"/>
<line id="rightarm2" x1="356" y1="290" x2="356" y2="330"
  style="stroke:black; stroke-width:20px; stroke-linecap:round"/>
</svg>
```

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

Nachname: Vorname:

Matr.Nr.: Studiengang:

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 120 Minuten.

Aufgabe	Punkte max.	Punkte erreicht
1	6	
2	2	
3	6	
4	6	
5	4	
6	4	
7	4	
Summe	**32**	

Die hohen Strompreise haben bis in diese Klausur durchgeschlagen. Betrachten Sie die folgende, fiktive Übersicht im Dokument **`Stromanbieter.xml`**.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="AnbieterStyle.xsl" ?>
<!DOCTYPE Stromanbieter SYSTEM "Stromanbieter.dtd">
<Stromanbieter Stand="2013-02-15" Kundengruppe="privat"
      GrundlageKWh="4000">
  <Angebot seit="2013-01-01">
    <Anbieter Tarif="GanzEasy">Wasserfall</Anbieter>
    <Preis>940.00</Preis>
    <Merkmale Vorkasse="nein" Kuendigungsfrist="1 Monat"
        Preisgarantie="12 Monate" Wechslerbonus="190.00"/>
  </Angebot>
  <Angebot seit="2013-02-13">
    <Anbieter Tarif="Sonnenpracht">rosastrom</Anbieter>
    <Preis>912.00</Preis>
    <Merkmale Vorkasse="ja" Kuendigungsfrist="12 Monate"
        Preisgarantie="12 Monate" Wechslerbonus="231.77"/>
  </Angebot>
  <Angebot seit="2013-02-10">
    <Anbieter Tarif="ClassicFix">Stadtwerke Musterstadt</Anbieter>
    <Preis>1002.34</Preis>
    <Merkmale Vorkasse="nein" Kuendigungsfrist="12 Monate"
        Preisgarantie="12 Monate" Wechslerbonus="75.00"/>
  </Angebot>
  <Angebot seit="2013-01-01">
    <Anbieter Tarif="ClassicFrei">Stadtwerke Musterstadt</Anbieter>
    <Preis>1032.34</Preis>
    <Merkmale Vorkasse="nein" Kuendigungsfrist="6 Wochen"
        Wechslerbonus="75.00"/>
  </Angebot>
</Stromanbieter>
```

Aufgabe 1:

Geben Sie eine DTD für das Stromanbieter-Dokument oben an. Es gelten die folgenden Vorgaben:

- *Stromanbieter* enthält mindestens ein *Angebot*.
- Alle Attribute außer den Aufzählungen sind vom Typ Zeichenkette.
- Die Angabe der Attribute *Stand* und *GrundlageKWh* ist verpflichtend, für das Attribut *Kundengruppe* ist nur „privat“ oder „geschaeftlich“ zulässig, mit „privat“ als Default-Wert.
- Ein *Angebot* besteht aus je einem Unterelement *Anbieter, Preis, Merkmale* in dieser Reihenfolge.
- Das Attribut *seit* in *Angebot* ist optional.
- Das Element *Anbieter* und das Element *Preis* haben, wie oben zu sehen, keine Unterelemente.
- Das Attribut *Tarif* in *Anbieter* ist Pflichtangabe.
- Das Element *Merkmale* darf weder Text noch Unterelemente enthalten.
- Die Deklaration der Attribute von *Merkmale* dürfen Sie weglassen.

```
<!-- DTD fuer die Stromanbieter -->
```

Aufgabe 2:

Kreuzen Sie die richtigen Aussagen an!

( ) In XML kommt es auf Groß- und Kleinschreibung an.

( ) Die Reihenfolge der Attribute eines Elements lässt sich erzwingen.

( ) Die Begriffe „valide“ und „wohlgeformt“ sind gleichwertig.

( ) Der Typ ID erzwingt eindeutige Attributwerte.

( ) Leere Elemente können keine Attribute haben.

( ) Textinhalt und Unterelemente können nicht gleichzeitig in einem Element auftreten.

Aufgabe 3:

Ein XML-Schema zum Stromanbieter-Dokument sieht wie folgt aus. Füllen Sie die Lücken. Orientieren Sie sich bei den gewünschten Angaben an den Vorgaben zur DTD aus Aufgabe 1. Zusätzlich bzw. abweichend soll gelten:

- Die Attribute *Stand* und *seit* haben einen geeigneten Datumstyp.
- Das Element *Preis* und das Attribut *Wechslerbonus* haben einen dezimalen Datentyp, das Attribut *GrundlageKWh* ist eine positive ganze Zahl.
- Das Attribut *Vorkasse* hat einen Aufzählungstyp mit den Werten „ja“, „nein“ und „k.A.“, letzterer ist Default-Wert.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:element name="Stromanbieter" type="_____________________"/>

  <xsd:complexType name="_____________________">
  <xsd:sequence>
    <xsd:element name="Angebot" type="TAngebot"
        maxOccurs="_____________________"/>
  </xsd:sequence>
  <xsd:attribute name="Stand" type="_____________" use="_________"/>
  <xsd:attribute name="Kundengruppe" default="privat">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="privat"/>
        <xsd:enumeration value="geschaeftlich"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
```

```
  <xsd:attribute name="GrundlageKWh" type="_____________________"
      use="_____________"/>
  </xsd:complexType>

  <xsd:complexType name="TAngebot">
    <xsd:sequence>
      <xsd:element name="Anbieter" type="TAnbieter"/>
      <xsd:element name="Preis" type="_________________"/>
      <xsd:element name="Merkmale" type="TMerkmale"/>
    </xsd:sequence>
    <xsd:attribute name="seit" type="_____________"/>
  </xsd:complexType>

  <xsd:complexType name="TAnbieter">
    <xsd:_________________>
      <xsd:extension base="_____________">
        <xsd:attribute name="Tarif" type="xsd:string"
            use="required"/>
      </xsd:extension>
    </xsd:_________________>
  </xsd:complexType>

  <xsd:complexType name="TMerkmale">
    <xsd:attribute name="Vorkasse" _____________________________>
      ____________________________________________________________
      ____________________________________________________________
      ____________________________________________________________
      ____________________________________________________________
      ____________________________________________________________
      ____________________________________________________________
      ____________________________________________________________
    </xsd:attribute>
    <xsd:attribute name="Kuendigungsfrist" type="xsd:string"
        use="required"/>
    <xsd:attribute name="Preisgarantie" type="xsd:string"/>
    <xsd:attribute name="Wechslerbonus" type="_________________"/>
  </xsd:complexType>

</xsd:schema>
```

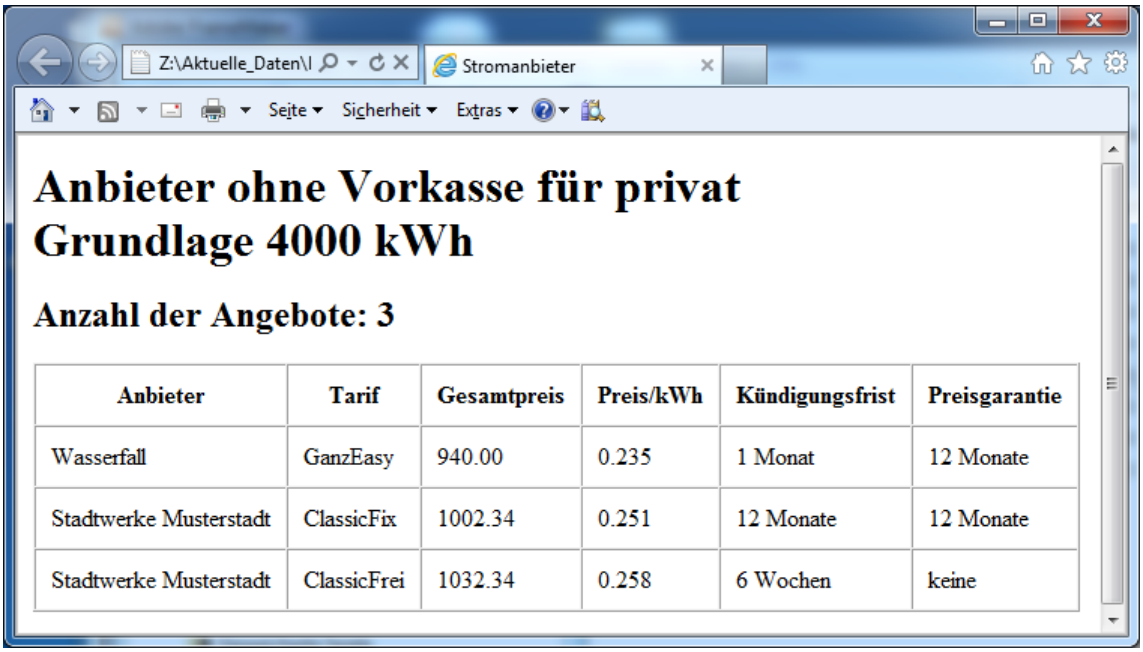
Aufgabe 4:

Ergänzen Sie die Lücken im Stylesheet, damit unser Stromanbieter-Dokument die unten gezeigte Ausgabe für Angebote mit Vorkasse „nein“ erzeugt.

```
<?xml version='1.0'?>
<xsl:stylesheet version='1.0'
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<head><title>Stromanbieter</title></head>
<body>
  <h1>
    Anbieter ohne Vorkasse für
    <xsl:value-of select="_____________________________"/><br/>
    Grundlage <xsl:value-of select="_____________________________"/>
    kWh
  </h1>
  <h2>
    Anzahl der Angebote: <xsl:value-of
      select="__________(//Angebot[_____________________________])"/>
  </h2>
  <p/>
  <table border="1" cellspacing="0" cellpadding="10">
    <tr>
      <th>Anbieter</th><th>Tarif</th><th>Gesamtpreis</th>
      <th>Preis/kWh</th><th>Kündigungsfrist</th><th>Preisgarantie</th>
    </tr>
    <xsl:_____________________
            select="//Angebot[_____________________________]"/>
  </table>
</body>
</html>
</xsl:template>

<xsl:template match="_____________________">
  <tr>
    <td><xsl:value-of select="Anbieter"/></td>
    <td><xsl:value-of select="_____________________________"/></td>
    <td><xsl:value-of select="Preis"/></td>
```

```
    <xsl:variable name="_____________________________"
          select="Preis div ../@GrundlageKWh"/>
    <td><xsl:value-of select="round($PreisProKWh * 1000) div 1000"/></td>
    <td><xsl:value-of select="Merkmale/@Kuendigungsfrist"/></td>
    <td>
      <xsl:_____________>
        <xsl:when test="Merkmale/@Preisgarantie">
          <xsl:value-of select="_____________________________________"/>
        </xsl:when>
        <xsl:otherwise>keine</xsl:otherwise>
      </xsl:_____________>
    </td>
  </tr>
</xsl:template>

</xsl:stylesheet>
```

Ausgabe:



Anbieter ohne Vorkasse für privat
Grundlage 4000 kWh

Anzahl der Angebote: 3

Anbieter	Tarif	Gesamtpreis	Preis/kWh	Kündigungsfrist	Preisgarantie
Wasserfall	GanzEasy	940.00	0.235	1 Monat	12 Monate
Stadtwerke Musterstadt	ClassicFix	1002.34	0.251	12 Monate	12 Monate
Stadtwerke Musterstadt	ClassicFrei	1032.34	0.258	6 Wochen	keine

Aufgabe 5:

Füllen Sie die Lücken in der folgenden XQuery auf dem Dokument **`Stromanbieter.xml`**, sodass das unten gezeigte Ergebnis geliefert wird?

```
let _____________________________________ :=
     fn:doc("Stromanbieter.xml")//Angebot[Merkmale/@Vorkasse = 'nein']
let $minPreis := fn:min($angOhneVk/(Preis - Merkmale/@Wechslerbonus))
for $x in $angOhneVk where ____________________________________________
     = $minPreis

_____________
     <BilligUndOhneVorkasse>
       <Firma>{fn:string($x/Anbieter)}</Firma>
       <Tarif>{fn:string(_____________________________________)}</Tarif>
       <EffektiverPreis>_____________________________________</EffektiverPreis>
     </BilligUndOhneVorkasse>
```

Ergebnis:

```
<BilligUndOhneVorkasse>
    <Firma>Wasserfall</Firma>
    <Tarif>GanzEasy</Tarif>
    <EffektiverPreis>750</EffektiverPreis>
</BilligUndOhneVorkasse>
```

Aufgabe 6:

Gegeben sei die folgende Datenbanktabelle mit Namen **STROMANBIETER**.

ANBIETER	TARIF	GPREIS	VKASSE	KFRIST	PGARANTIE	WBONUS
Wasserfall	GanzEasy	940.00	nein	1 Monat	12 Monate	190.00
rosastrom	Sonnen-pracht	912.00	ja	12 Monate	12 Monate	231.77
Stadtwerke Musterstadt	ClassicFix	1002.34	nein	12 Monate	12 Monate	75.00
Stadtwerke Musterstadt	ClassicFrei	1032.34	nein	6 Wochen	*NULL*	75.00

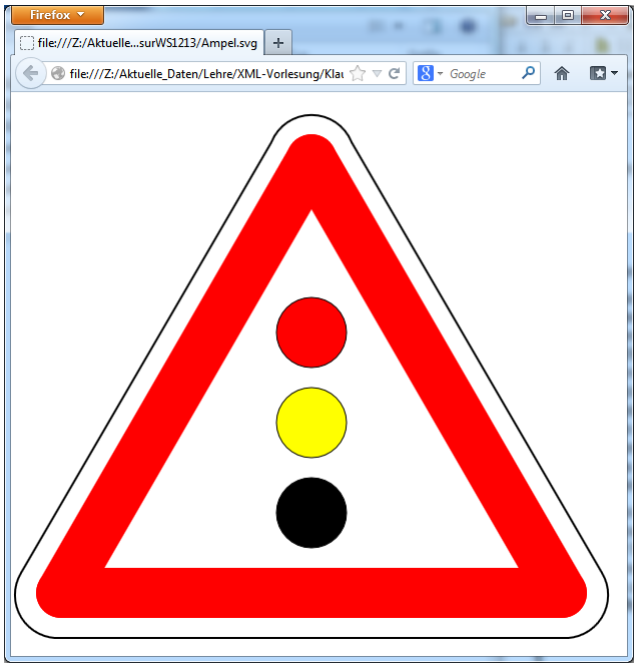
Füllen Sie die Lücken in der SQL/XML-Abfrage auf der STROMANBIETER-Tabelle, die als Ergebnis die unten gezeigte Ausgabe liefert. In der Ausgabe sind jetzt alle Tarife Unterelemente des zugehörigen Anbieters.

```
SELECT XMLELEMENT (NAME "Anbieter",
            XMLATTRIBUTES (ANBIETER AS "Firma"),
            _____________________________________
            _____________________________________
            _____________________________________
            _____________________________________
            _____________________________________
)
FROM STROMANBIETER
GROUP BY ANBIETER;
```

Ausgabe:

```
<Anbieter Firma="Stadtwerke Musterstadt">
  <Tarif Bezeichnung="ClassicFix"/>
  <Tarif Bezeichnung="ClassicFrei"/>
</Anbieter>
<Anbieter Firma="Wasserfall">
  <Tarif Bezeichnung="GanzEasy"/>
</Anbieter>
<Anbieter Firma="rosastrom">
  <Tarif Bezeichnung="Sonnenpracht"/>
</Anbieter>
```

Aufgabe 7:



In unserer Sammlung animierter Verkehrszeichen betrachten wir heute das Gefahrenzeichen 131 *Lichtzeichenanlage*. Eine Ampel hat bekanntlich vier Phasen: rot, rot-gelb, grün, gelb. In unserer Animation soll jede Phase 2 Sekunden dauern, nichtleuchtende „Lampen“ werden schwarz dargestellt.

Ergänzen Sie die Lücken!

```
<?xml version="1.0"?>
<svg xmlns=
   "http://www.w3.org/2000/svg">
<rect x="0" y="0" width="600"
   height="600" fill="white"/>
<line x1="50" y1="500"
   x2="550" y2="500"
   style="stroke:red; stroke-width:50px; stroke-linecap:round" />
<line x1="50" y1="500" x2="300" y2="67"
   style="stroke:red; stroke-width:50px; stroke-linecap:round" />
<line x1="300" y1="67" x2="550" y2="500"
   style="stroke:red; stroke-width:50px; stroke-linecap:round" />
<path style="stroke:black; stroke-width:2px" fill="none"
   d="M 260 50 A 43 43 60 0 1 340 50 L 590 480
      A 43 43 -60 0 1 550 545 L 50 545 A 43 43 -60 0 1 10 480 L 260 50" />

   <circle id="rot" cx="300" cy="240" r="35" fill="red" stroke="black">
      <animate attributeName="_____________________"
         values="_____________________________________________________"
         calcMode="discrete" dur="________" repeatCount="indefinite"/>
   </circle>
   <circle id="gelb" cx="300" cy="330" r="35" fill="black" stroke="black">
      <animate attributeName="_____________________"
         values="black; yellow; black; yellow"
         calcMode="discrete" dur="________" repeatCount="indefinite"/>
   </circle>
   <circle id="gruen" cx="300" cy="420" r="35" fill="black" stroke="black">
      <animate attributeName="_____________________"
         values="_____________________________________________________"
         calcMode="discrete" dur="________" repeatCount="indefinite"/>
   </circle>

</svg>
```

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

Nachname: Vorname:

Matr.Nr.: Studiengang:

MUSTERLÖSUNG

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 120 Minuten.

Aufgabe	Punkte max.	Punkte erreicht
1	6	
2	2	
3	6	
4	6	
5	4	
6	4	
7	4	
Summe	**32**	

Die hohen Strompreise haben bis in diese Klausur durchgeschlagen. Betrachten Sie die folgende, fiktive Übersicht im Dokument **`Stromanbieter.xml`**.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="AnbieterStyle.xsl" ?>
<!DOCTYPE Stromanbieter SYSTEM "Stromanbieter.dtd">
<Stromanbieter Stand="2013-02-15" Kundengruppe="privat"
      GrundlageKWh="4000">
  <Angebot seit="2013-01-01">
    <Anbieter Tarif="GanzEasy">Wasserfall</Anbieter>
    <Preis>940.00</Preis>
    <Merkmale Vorkasse="nein" Kuendigungsfrist="1 Monat"
        Preisgarantie="12 Monate" Wechslerbonus="190.00"/>
  </Angebot>
  <Angebot seit="2013-02-13">
    <Anbieter Tarif="Sonnenpracht">rosastrom</Anbieter>
    <Preis>912.00</Preis>
    <Merkmale Vorkasse="ja" Kuendigungsfrist="12 Monate"
        Preisgarantie="12 Monate" Wechslerbonus="231.77"/>
  </Angebot>
  <Angebot seit="2013-02-10">
    <Anbieter Tarif="ClassicFix">Stadtwerke Musterstadt</Anbieter>
    <Preis>1002.34</Preis>
    <Merkmale Vorkasse="nein" Kuendigungsfrist="12 Monate"
        Preisgarantie="12 Monate" Wechslerbonus="75.00"/>
  </Angebot>
  <Angebot seit="2013-01-01">
    <Anbieter Tarif="ClassicFrei">Stadtwerke Musterstadt</Anbieter>
    <Preis>1032.34</Preis>
    <Merkmale Vorkasse="nein" Kuendigungsfrist="6 Wochen"
        Wechslerbonus="75.00"/>
  </Angebot>
</Stromanbieter>
```

Aufgabe 1:

Geben Sie eine DTD für das Stromanbieter-Dokument oben an. Es gelten die folgenden Vorgaben:

- *Stromanbieter* enthält mindestens ein *Angebot.*
- Alle Attribute außer den Aufzählungen sind vom Typ Zeichenkette.
- Die Angabe der Attribute *Stand* und *GrundlageKWh* ist verpflichtend, für das Attribut *Kundengruppe* ist nur „privat“ oder „geschaeftlich“ zulässig, mit „privat“ als Default-Wert.
- Ein *Angebot* besteht aus je einem Unterelement *Anbieter, Preis, Merkmale* in dieser Reihenfolge.
- Das Attribut *seit* in *Angebot* ist optional.
- Das Element *Anbieter* und das Element *Preis* haben, wie oben zu sehen, keine Unterelemente.
- Das Attribut *Tarif* in *Anbieter* ist Pflichtangabe.
- Das Element *Merkmale* darf weder Text noch Unterelemente enthalten.
- Die Deklaration der Attribute von *Merkmale* dürfen Sie weglassen.

```
<!-- DTD fuer die Sromangebote -->
<!ELEMENT Stromanbieter (Angebot+)>
<!ATTLIST Stromanbieter Stand CDATA #REQUIRED
                 Kundengruppe (privat | geschaeftlich) "privat"
                 GrundlageKWh CDATA #REQUIRED>

<!ELEMENT Angebot (Anbieter, Preis, Merkmale)>
<!ATTLIST Angebot seit CDATA #IMPLIED>

<!ELEMENT Anbieter (#PCDATA)>
<!ATTLIST Anbieter Tarif CDATA #REQUIRED>

<!ELEMENT Preis (#PCDATA)>

<!ELEMENT Merkmale EMPTY>
<!ATTLIST Merkmale Vorkasse (ja | nein | k.A.) "k.A."
                   Kuendigungsfrist CDATA #REQUIRED
                   Preisgarantie CDATA #IMPLIED
                   Wechslerbonus CDATA #IMPLIED>
```

war nicht verlangt

Aufgabe 2:

Kreuzen Sie die richtigen Aussagen an!

(X) In XML kommt es auf Groß- und Kleinschreibung an.

( ) Die Reihenfolge der Attribute eines Elements lässt sich erzwingen.

( ) Die Begriffe „valide“ und „wohlgeformt“ sind gleichwertig.

(X) Der Typ ID erzwingt eindeutige Attributwerte.

(*) ( ) Leere Elemente können keine Attribute haben.

( ) Textinhalt und Unterelemente können nicht gleichzeitig in einem Element auftreten.

(*) Alternative nicht gewertet, weil mehrdeutig

Aufgabe 3:

Ein XML-Schema zum Stromanbieter-Dokument sieht wie folgt aus. Füllen Sie die Lücken. Orientieren Sie sich bei den gewünschten Angaben an den Vorgaben zur DTD aus Aufgabe 1. Zusätzlich bzw. abweichend soll gelten:

- Die Attribute *Stand* und *seit* haben einen geeigneten Datumstyp.
- Das Element *Preis* und das Attribut *Wechslerbonus* haben einen dezimalen Datentyp, das Attribut *GrundlageKWh* ist eine positive ganze Zahl.
- Das Attribut *Vorkasse* hat einen Aufzählungstyp mit den Werten „ja“, „nein“ und „k.A.“, letzterer ist Default-Wert.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:element name="Stromanbieter" type="_TStromanbieter_"/>

  <xsd:complexType name="_TStromanbieter_">
  <xsd:sequence>
    <xsd:element name="Angebot" type="TAngebot"
        maxOccurs="_unbounded_"/>
  </xsd:sequence>
  <xsd:attribute name="Stand" type="_xsd:date_" use="_required_"/>
  <xsd:attribute name="Kundengruppe" default="privat">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="privat"/>
        <xsd:enumeration value="geschaeftlich"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
```

```
  <xsd:attribute name="GrundlageKWh" type="_xsd:positiveInteger_"
      use="_required_"/>
  </xsd:complexType>

  <xsd:complexType name="TAngebot">
    <xsd:sequence>
      <xsd:element name="Anbieter" type="TAnbieter"/>
      <xsd:element name="Preis" type="_xsd:decimal_"/>
      <xsd:element name="Merkmale" type="TMerkmale"/>
    </xsd:sequence>
    <xsd:attribute name="seit" type="_xsd:date_"/>
  </xsd:complexType>

  <xsd:complexType name="TAnbieter">
    <xsd:_simpleContent_>
      <xsd:extension base="_xsd:string_">
        <xsd:attribute name="Tarif" type="xsd:string"
            use="required"/>
      </xsd:extension>
    </xsd:_simpleContent_>
  </xsd:complexType>

  <xsd:complexType name="TMerkmale">
    <xsd:attribute name="Vorkasse" _default="k.A."_>
      _<xsd:simpleType>_
      __<xsd:restriction base="xsd:string">__
      _____<xsd:enumeration value="ja"/>__
      _____<xsd:enumeration value="nein"/>___
      _____<xsd:enumeration value="k.A."/>___
      __</xsd:restriction>__
      _</xsd:simpleType>_
    </xsd:attribute>
    <xsd:attribute name="Kuendigungsfrist" type="xsd:string"
        use="required"/>
    <xsd:attribute name="Preisgarantie" type="xsd:string"/>
    <xsd:attribute name="Wechslerbonus" type="__xsd:decimal__"/>
  </xsd:complexType>

</xsd:schema>
```

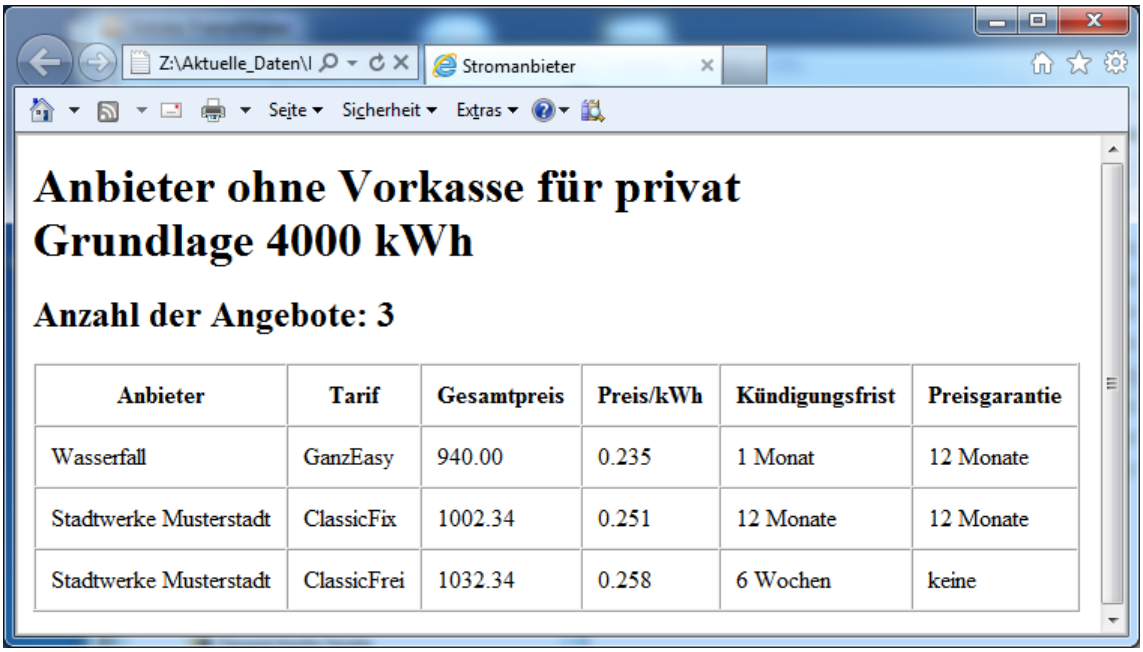
Aufgabe 4:

Ergänzen Sie die Lücken im Stylesheet, damit unser Stromanbieter-Dokument die unten gezeigte Ausgabe für Angebote mit Vorkasse „nein“ erzeugt.

```
<?xml version='1.0'?>
<xsl:stylesheet version='1.0'
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<head><title>Stromanbieter</title></head>
<body>
  <h1>
    Anbieter ohne Vorkasse für
    <xsl:value-of select="__Stromanbieter/@Kundengruppe__"/><br/>
    Grundlage <xsl:value-of select="__Stromanbieter/@GrundlageKWh__"/>
    kWh
  </h1>
  <h2>
    Anzahl der Angebote: <xsl:value-of
      select="_count_(//Angebot[_Merkmale/@Vorkasse='nein'_])"/>
  </h2>
  <p/>
  <table border="1" cellspacing="0" cellpadding="10">
    <tr>
      <th>Anbieter</th><th>Tarif</th><th>Gesamtpreis</th>
      <th>Preis/kWh</th><th>Kündigungsfrist</th><th>Preisgarantie</th>
    </tr>
    <xsl:__apply-templates__
            select="//Angebot[_Merkmale/@Vorkasse='nein'_]"/>
  </table>
</body>
</html>
</xsl:template>

<xsl:template match="__Angebot__">
  <tr>
    <td><xsl:value-of select="Anbieter"/></td>
    <td><xsl:value-of select="__Anbieter/@Tarif__"/></td>
    <td><xsl:value-of select="Preis"/></td>
```

```
    <xsl:variable name="__PreisProKWh__"
          select="Preis div ../@GrundlageKWh"/>
    <td><xsl:value-of select="round($PreisProKWh * 1000) div 1000"/></td>
    <td><xsl:value-of select="Merkmale/@Kuendigungsfrist"/></td>
    <td>
      <xsl:_choose__>
        <xsl:when test="Merkmale/@Preisgarantie">
          <xsl:value-of select="__Merkmale/@Preisgarantie__"/>
        </xsl:when>
        <xsl:otherwise>keine</xsl:otherwise>
      </xsl:_choose_>
    </td>
  </tr>
</xsl:template>

</xsl:stylesheet>
```

Ausgabe:



Anbieter ohne Vorkasse für privat
Grundlage 4000 kWh

Anzahl der Angebote: 3

Anbieter	Tarif	Gesamtpreis	Preis/kWh	Kündigungsfrist	Preisgarantie
Wasserfall	GanzEasy	940.00	0.235	1 Monat	12 Monate
Stadtwerke Musterstadt	ClassicFix	1002.34	0.251	12 Monate	12 Monate
Stadtwerke Musterstadt	ClassicFrei	1032.34	0.258	6 Wochen	keine

Aufgabe 5:

Füllen Sie die Lücken in der folgenden XQuery auf dem Dokument **`Stromanbieter.xml`**, sodass das unten gezeigte Ergebnis geliefert wird?

```
let __$angOhneVk__ :=
     fn:doc("Stromanbieter.xml")//Angebot[Merkmale/@Vorkasse = 'nein']
let $minPreis := fn:min($angOhneVk/(Preis - Merkmale/@Wechslerbonus))
for $x in $angOhneVk where __$x/Preis - $x/Merkmale/@Wechslerbonus__
     = $minPreis                                                    (*)
_return_
     <BilligUndOhneVorkasse>
       <Firma>{fn:string($x/Anbieter)}</Firma>
       <Tarif>{fn:string(__$x/Anbieter/@Tarif__)}</Tarif>
       <EffektiverPreis>__{$minPreis}__</EffektiverPreis>
     </BilligUndOhneVorkasse>
```

Ergebnis:

```
<BilligUndOhneVorkasse>
    <Firma>Wasserfall</Firma>
    <Tarif>GanzEasy</Tarif>
    <EffektiverPreis>750</EffektiverPreis>
</BilligUndOhneVorkasse>
```

(*) auch $x/(Preis - Merkmale/@Wechslerbonus) möglich

Aufgabe 6:

Gegeben sei die folgende Datenbanktabelle mit Namen **STROMANBIETER**.

ANBIETER	TARIF	GPREIS	VKASSE	KFRIST	PGARANTIE	WBONUS
Wasserfall	GanzEasy	940.00	nein	1 Monat	12 Monate	190.00
rosastrom	Sonnen-pracht	912.00	ja	12 Monate	12 Monate	231.77
Stadtwerke Musterstadt	ClassicFix	1002.34	nein	12 Monate	12 Monate	75.00
Stadtwerke Musterstadt	ClassicFrei	1032.34	nein	6 Wochen	*NULL*	75.00

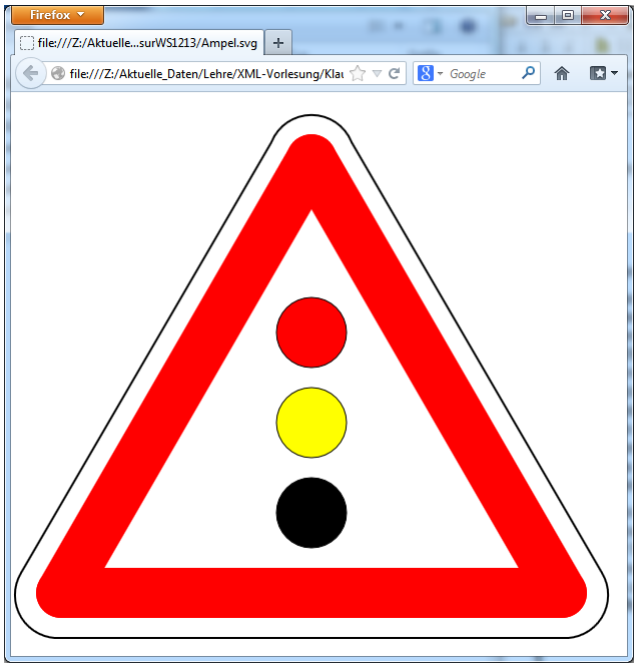
Füllen Sie die Lücken in der SQL/XML-Abfrage auf der STROMANBIETER-Tabelle, die als Ergebnis die unten gezeigte Ausgabe liefert. In der Ausgabe sind jetzt alle Tarife Unterelemente des zugehörigen Anbieters.

```
SELECT XMLELEMENT (NAME "Anbieter",
            XMLATTRIBUTES (ANBIETER AS "Firma"),
            __XMLAGG( __
            ____XMLELEMENT (NAME "Tarif", ___
            _______XMLATTRIBUTES (TARIF AS "Bezeichnung") __
            ____ ) __
            _ ) __
)
FROM STROMANBIETER
GROUP BY ANBIETER;
```

Ausgabe:

```
<Anbieter Firma="Stadtwerke Musterstadt">
  <Tarif Bezeichnung="ClassicFix"/>
  <Tarif Bezeichnung="ClassicFrei"/>
</Anbieter>
<Anbieter Firma="Wasserfall">
  <Tarif Bezeichnung="GanzEasy"/>
</Anbieter>
<Anbieter Firma="rosastrom">
  <Tarif Bezeichnung="Sonnenpracht"/>
</Anbieter>
```

Aufgabe 7:



In unserer Sammlung animierter Verkehrszeichen betrachten wir heute das Gefahrenzeichen 131 *Lichtzeichenanlage*. Eine Ampel hat bekanntlich vier Phasen: rot, rot-gelb, grün, gelb. In unserer Animation soll jede Phase 2 Sekunden dauern, nichtleuchtende „Lampen“ werden schwarz dargestellt.

Ergänzen Sie die Lücken!

```
<?xml version="1.0"?>
<svg xmlns=
   "http://www.w3.org/2000/svg">
<rect x="0" y="0" width="600"
   height="600" fill="white"/>
<line x1="50" y1="500"
   x2="550" y2="500"
   style="stroke:red; stroke-width:50px; stroke-linecap:round" />
<line x1="50" y1="500" x2="300" y2="67"
   style="stroke:red; stroke-width:50px; stroke-linecap:round" />
<line x1="300" y1="67" x2="550" y2="500"
   style="stroke:red; stroke-width:50px; stroke-linecap:round" />
<path style="stroke:black; stroke-width:2px" fill="none"
   d="M 260 50 A 43 43 60 0 1 340 50 L 590 480
      A 43 43 -60 0 1 550 545 L 50 545 A 43 43 -60 0 1 10 480 L 260 50" />

   <circle id="rot" cx="300" cy="240" r="35" fill="red" stroke="black">
      <animate attributeName="__fill__"
         values="_red; red; black; black_"
         calcMode="discrete" dur="_8s_" repeatCount="indefinite"/>
   </circle>
   <circle id="gelb" cx="300" cy="330" r="35" fill="black" stroke="black">
      <animate attributeName="__fill__"
         values="black; yellow; black; yellow"
         calcMode="discrete" dur="__8s__" repeatCount="indefinite"/>
   </circle>
   <circle id="gruen" cx="300" cy="420" r="35" fill="black" stroke="black">
      <animate attributeName="__fill__"
         values="_black; black; green; black_"
         calcMode="discrete" dur="__8s__" repeatCount="indefinite"/>
   </circle>

</svg>
```

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

Nachname: Vorname:

Matr.Nr.: Studiengang:

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 90 Minuten.

Aufgabe	Punkte max.	Punkte erreicht
1	2 + 2	
2	4	
3	2	
4	5	
5	4	
6	4	
7	4	
8	5	
Summe	**32**	

Aufgabe 1:

Das „Jahrhunderthochwasser“ im Frühsommer hat sich bis in diese Klausur durchgeschlagen. Aber keine Sorge, wir machen Sie nicht nass. Betrachten Sie die folgende, fiktive Pegelstandsmeldung **`Pegel.xml`** der Werra bei Allendorf.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="PegelStyle.xsl" ?>
<Pegel Name="Allendorf" Mst_Nr="41900104" Flussgebiet="Werra">
  <Stammdaten>
    <Einzugsgebiet Einheit="qkm">5166</Einzugsgebiet>
    <Entfernung_Muendung Einheit="km">40.7</Entfernung_Muendung>
    <Betreiber>WSV</Betreiber>
    <Hochwassermeldepegel>ja</Hochwassermeldepegel>
    <Meldestufe1 Einheit="cm">310</Meldestufe1>
    <Meldestufe2 Einheit="cm">380</Meldestufe2>
    <Meldestufe3 Einheit="cm">450</Meldestufe3>
    <HHW wann="26.03.1987" Einheit="cm">434</HHW>
  </Stammdaten>
  <Wasserstand Einheit="cm">
    <Wert W="425" T="06.06.2013" U="12:00"/>
  </Wasserstand>
</Pegel>
```

(a) Ist das Dokument ein wohlgeformtes XML-Dokument? Wenn nein, welche Fehler enthält es?

(b) Könnte man alle Unterelemente von **`<Stammdaten>`** in leere Elemente umwandeln, indem man die Inhalte zu Attributen wandelt? Wenn ja, wie würde z.B. das Element **`<Betreiber>`** aussehen? Wenn nein, warum nicht?

Aufgabe 2:

Ergänzen Sie die unterstrichenen Lücken in der DTD für das (ggf. korrigierte) Pegel-Dokument von oben! Setzen Sie dazu die richtigen Ziffern zu den vorgeschlagenen Ergänzungen ein. Es gilt die folgende Vorgabe: Das Attribut *Flussgebiet* ist optional, alle anderen Attribute sind Pflichtangaben. Achten Sie auf leere Elemente! Hinweis: Nicht alle Ergänzungen werden gebraucht, manche aber mehrfach.

1 #OPTIONAL
2 #IMPLIED
3 (#PCDATA)
4 (PCDATA)
5 #PCDATA
6 EMPTY
7 #EMPTY

```
<!-- DTD fuer den Pegel -->
<!ELEMENT Pegel (Stammdaten, Wasserstand)>
<!ATTLIST Pegel  Name CDATA #REQUIRED
                 Mst_Nr CDATA #REQUIRED
                 Flussgebiet CDATA _______>

<!ELEMENT Stammdaten (Einzugsgebiet,
       Entfernung_Muendung, Betreiber,
       Hochwassermeldepegel, Meldestufe1,
       Meldestufe2, Meldestufe3, HHW)>

<!ELEMENT Einzugsgebiet ________>
<!ATTLIST Einzugsgebiet Einheit CDATA #REQUIRED>

<!ELEMENT Entfernung_Muendung _______>
<!ATTLIST Entfernung_Muendung Einheit CDATA #REQUIRED>

<!ELEMENT Betreiber _______>

<!ELEMENT Hochwassermeldepegel _______>

<!ELEMENT Meldestufe1 _______>
<!ATTLIST Meldestufe1 Einheit CDATA #REQUIRED>
...
<!ELEMENT HHW _______>
<!ATTLIST HHW  wann CDATA #REQUIRED
               Einheit CDATA #REQUIRED>

<!ELEMENT Wasserstand (Wert)>
<!ATTLIST Wasserstand Einheit CDATA #REQUIRED>

<!ELEMENT Wert ___________>
<!ATTLIST Wert W CDATA #REQUIRED
               T CDATA #REQUIRED
               U CDATA #REQUIRED>
```

Aufgabe 3:

Nehmen wir an, das Element *Hochwassermeldepegel* wäre jetzt ein leeres Element mit dem Attribut *A*, das nur die Werte *ja, nein, k.A.* annehmen darf, wobei *k.A.* der Default-Wert ist. Ergänzen Sie für diese neue Vorschrift die Lücken in der DTD-Angabe unten.

```
<!ELEMENT Hochwassermeldepegel _____________________________________>

<!ATTLIST Hochwassermeldepegel A ____________________________________>
```

Aufgabe 4:

Ein XML-Schema zum Pegel-Dokument sieht wie folgt aus. Füllen Sie die Lücken. Orientieren Sie sich bei den gewünschten Angaben an den Vorgaben der DTD aus Aufgabe 2. Zusätzlich bzw. abweichend soll gelten:

- Das Attribut *W* im Element *Wert* ist eine positive ganze Zahl. Die Attribute *T* und *U* seien als Zeichenketten deklariert.
- Die Inhalte der Elemente *Einzugsgebiet, Entfernung_Muendung,* die *Meldestufen* und *HHW* seien als Dezimalzahlen deklariert.
- Alle Attribute *Einheit* seien vom Typ Zeichenkette.
- Beachten Sie, dass das Attribut *Flussgebiet* weiterhin optional ist.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Pegel" type="TPegel"/>
  <xsd:complexType name="TPegel">
    <_________________>
      <xsd:element name="Stammdaten" type="____________________"/>
      <xsd:element name="Wasserstand" type="TWasserstand"/>
    </_________________>
    <xsd:attribute name="Name" type="xsd:string" use="required"/>
    <xsd:attribute name="Mst_Nr" type="xsd:integer" use="required"/>
    <xsd:attribute name="Flussgebiet" type="xsd:string"
          use="____________________"/>
  </xsd:complexType>

  <xsd:complexType name="____________________">
    <xsd:sequence>
      <xsd:element name="Einzugsgebiet" type="_____________"/>
      <xsd:element name="Entfernung_Muendung" type="_____________"/>
      <xsd:element name="Betreiber" type="xsd:string"/>
      <xsd:element name="Hochwassermeldepegel" type="xsd:string"/>
      <xsd:element name="Meldestufe1" type="TEinheit"/>
```

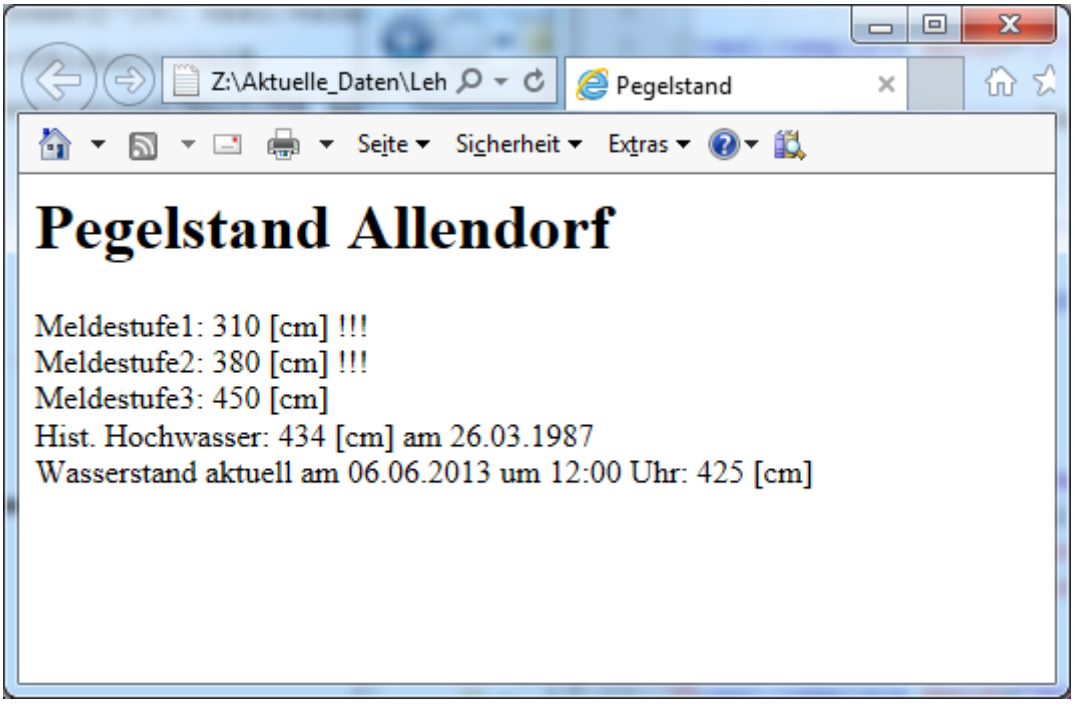
```
      <xsd:element name="Meldestufe2" type="TEinheit"/>
      <xsd:element name="Meldestufe3" type="TEinheit"/>
      <xsd:element name="HHW" type="_____"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="TEinheit">
    <xsd:simpleContent>
      <xsd:extension base="xsd:decimal">
        <xsd:attribute name="Einheit" type="xsd:string"
              use="required"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>

  <xsd:complexType name="TEinheitWann">
    <xsd:simpleContent>
      <xsd:extension base="_____">
        <xsd:attribute name="wann" type="xsd:string"
              use="required"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>

  <xsd:complexType name="TWasserstand">
    <xsd:sequence>
      <xsd:element name="Wert">
        <xsd:complexType>
          <xsd:attribute name="W" type="_____"
                use="required"/>
          <xsd:attribute name="T" type="xsd:string" use="required"/>
          <xsd:attribute name="U" type="xsd:string" use="required"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="_____" type="xsd:string"
          use="required"/>
  </xsd:complexType>
</xsd:schema>
```

Aufgabe 5:

Ergänzen Sie die Lücken im Stylesheet, damit unser, ggf. korrigiertes Pegel-Dokument aus Aufgabe 1 die unten gezeigte Ausgabe erzeugt. Übertrifft ein aktueller Pegelstand eine oder mehrere Meldestufen, sollen drei Ausrufezeichen hinter dem jeweiligen Meldestufenwert erscheinen.

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet version='1.0'
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
    <head><title>Pegelstand</title></head>
    <body>
      <h1>Pegelstand
        <xsl:value-of select="_____________________________"/>
      </h1>
      <xsl:apply-templates select="Pegel/*"/>
    </body>
  </html>
</xsl:template>

<xsl:template match="_____________________________">
  <xsl:__________________ select="Meldestufe1|Meldestufe2|Meldestufe3"/>
  Hist. Hochwasser: <xsl:value-of select="HHW"/>
  [<xsl:value-of select="_____________________"/>]
  am <xsl:value-of select="_____________________"/><br/>
</xsl:template>

<xsl:template match="Meldestufe1|Meldestufe2|Meldestufe3">
  <xsl:value-of select="_______________"/>: <xsl:value-of select="."/>
  [<xsl:value-of select="@Einheit"/>]
  <xsl:if test="../../Wasserstand/Wert/@W &gt; _______"> !!!</xsl:if>
  <br/>
</xsl:template>

<xsl:template match="_____________________________">
  Wasserstand aktuell am <xsl:value-of select="Wert/@T"/> um
  <xsl:value-of select="Wert/@U"/> Uhr:
  <xsl:value-of select="Wert/@W"/> [<xsl:value-of select="@Einheit"/>]
</xsl:template>
</xsl:stylesheet>
```

Ausgabe:



Aufgabe 6:

Füllen Sie die Lücken in der folgenden XQuery auf dem ggf. korrigierten Dokument **`Pegel.xml`**, sodass das unten gezeigte Ergebnis (also der Pegelstand und die höchste überschrittene Meldestufe) geliefert wird?

```
let $p := fn:doc("Pegel.xml")/Pegel
let $m := ($p/Stammdaten/Meldestufe3[$p/Wasserstand/Wert/@W > .],
           $p/Stammdaten/Meldestufe2[$p/Wasserstand/Wert/@W > .],
           $p/Stammdaten/Meldestufe1[$p/Wasserstand/Wert/@W > .],
           <Meldestufen_nicht/>)
let $x := $m[1]
return
```

_____

_____

_____

_____

Ausgabe:

```
<Pegelmeldung Name="Allendorf">
    <Pegelstand W="425"/>
    <Warnung>Meldestufe2 ueberschritten</Warnung>
</Pegelmeldung>
```

Aufgabe 7:

Gegeben sei die folgende Datenbanktabelle mit Namen **PEGELSTAENDE**.

GEWAESSER	STATION	WASSER-STAND
Werra	Allendorf	139
Werra	Heldra	203
Weser	Bad Karlshafen	160
Lahn	Marburg	173
Fulda	Bad Hersfeld	237
Fulda	Rotenburg	165

Füllen Sie die Lücken in der SQL/XML-Abfrage auf der PEGELSTAENDE-Tabelle, die als Ergebnis die unten gezeigte Ausgabe liefert.

```
SELECT XMLELEMENT (NAME "Flussgebiet",
           XMLATTRIBUTES (GEWAESSER AS "Fluss"),
                ____________________________________________
                ____________________________________________
                ____________________________________________
                ____________________________________________
                ____________________________________________
                ____________________________________________
)
FROM PEGELSTAENDE
GROUP BY GEWAESSER;
```

Ausgabe:

```
<Flussgebiet Fluss="Werra">
  <Station Name="Allendorf">139</Station>
  <Station Name="Heldra">203</Station>
</Flussgebiet>
<Flussgebiet Fluss="Weser">
  <Station Name="Bad Karlshafen">160</Station>
</Flussgebiet>
...
```

Aufgabe 8:

In unserer Sammlung animierter Verkehrszeichen betrachten wir heute das Gefahrenzeichen 129 *Ufer*, das allerdings zum 01.04.2013 aus dem Regelkatalog gestrichen wurde.

Ergänzen Sie die Lücken durch Angabe der passenden Ziffer! Hinweis: Nicht alle Ergänzungen werden gebraucht, manche aber mehrfach.

```
1 circle
2 drive
3 fill
4 line
5 path
6 polygon
7 wave
```


```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg">
<rect x="0" y="0" width="600" height="600" fill="white"/>
<______ x1="50" y1="500" x2="550" y2="500"
  style="stroke:red; stroke-width:50px; stroke-linecap:round" />
<______ x1="50" y1="500" x2="300" y2="67"
  style="stroke:red; stroke-width:50px; stroke-linecap:round" />
<______ x1="300" y1="67" x2="550" y2="500"
  style="stroke:red; stroke-width:50px; stroke-linecap:round" />
<______ style="stroke:black; stroke-width:2px" fill="none"
  d="M 260 50 A 43 43 60 0 1 340 50 L 590 480
    A 43 43 -60 0 1 550 545 L 50 545
    A 43 43 -60 0 1 10 480 L 260 50" />

<______ points="110,465 168,368 250,368 250,465"
  style="fill:black;stroke:none"/>

<g id="car">
  <______ points="160,345 160,315 200,280 280,280 310,310 360,320 360,345"
    style="fill:black;stroke:none"/>
  <______ id="fensterhinten" points="190,310 210,285 238,285 238,310"
    style="fill:white;stroke:none"/>
  <______ id="fenstervorne" points="245,310 245,285 275,285 300,310"
    style="fill:white;stroke:none"/>
```

```
  <_____ id="wheel1" cx="200" cy="345" r="20"
    style="fill:black; stroke:white; stroke-width:4px"/>
  <_____ id="wheel2" cx="320" cy="345" r="20"
    style="fill:black; stroke:white; stroke-width:4px"/>
  <animateTransform attributeName="transform" type="rotate"
    from="6 220 600" to="22 230 600" begin="0s" dur="5s"
    repeatCount="indefinite"/>
</g>

<_____ style="stroke:black; stroke-width:15px" fill="none"
  d="M 250,420 q 25,-15 50,0 t 50,0" />
<_____ style="stroke:black; stroke-width:15px" fill="none"
  d="M 350,420 q 25,-15 50,0 t 50,0" />
<_____ style="stroke:black; stroke-width:15px" fill="none"
  d="M 250,450 q 25,-15 50,0 t 50,0" />
<_____ style="stroke:black; stroke-width:15px" fill="none"
  d="M 350,450 q 25,-15 50,0 t 50,0" />
<_____ style="stroke:black; stroke-width:15px" fill="none"
  d="M 449,451 q 15,-6 17, -7" />
<line style="stroke:white; stroke-width:10px" x1="255" y1="380" x2="255" y2="460"/>
</svg>
```

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

MUSTERLÖSUNG

Nachname: Vorname:

Matr.Nr.: Studiengang:

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 90 Minuten.

Aufgabe	Punkte max.	Punkte erreicht
1	2 + 2	
2	4	
3	2	
4	5	
5	4	
6	4	
7	4	
8	5	
Summe	**32**	

Aufgabe 1:

Das „Jahrhunderthochwasser“ im Frühsommer hat sich bis in diese Klausur durchgeschlagen. Aber keine Sorge, wir machen Sie nicht nass. Betrachten Sie die folgende, fiktive Pegelstandsmeldung **`Pegel.xml`** der Werra bei Allendorf.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="PegelStyle.xsl" ?>
<Pegel Name="Allendorf" Mst_Nr="41900104" Flussgebiet="Werra">
  <Stammdaten>
    <Einzugsgebiet Einheit="qkm">5166</Einzugsgebiet>
    <Entfernung_Muendung Einheit="km">40.7</Entfernung_Muendung>
    <Betreiber>WSV</Betreiber>
    <Hochwassermeldepegel>ja</Hochwassermeldepegel>
    <Meldestufe1 Einheit="cm">310</Meldestufe1>
    <Meldestufe2 Einheit="cm">380</Meldestufe2>
    <Meldestufe3 Einheit="cm">450</Meldestufe3>
    <HHW wann="26.03.1987" Einheit="cm">434</HHW>
  </Stammdaten>
  <Wasserstand Einheit="cm">
    <Wert W="425" T="06.06.2013" U="12:00"/>
  </Wasserstand>
</Pegel>
```

(a) Ist das Dokument ein wohlgeformtes XML-Dokument? Wenn nein, welche Fehler enthält es?

Ja

(b) Könnte man alle Unterelemente von **`<Stammdaten>`** in leere Elemente umwandeln, indem man die Inhalte zu Attributen wandelt? Wenn ja, wie würde z.B. das Element **`<Betreiber>`** aussehen? Wenn nein, warum nicht?

Ja. `<Betreiber Wert="WSV"/>`

Aufgabe 2:

Ergänzen Sie die unterstrichenen Lücken in der DTD für das (ggf. korrigierte) Pegel-Dokument von oben! Setzen Sie dazu die richtigen Ziffern zu den vorgeschlagenen Ergänzungen ein. Es gilt die folgende Vorgabe: Das Attribut *Flussgebiet* ist optional, alle anderen Attribute sind Pflichtangaben. Achten Sie auf leere Elemente! Hinweis: Nicht alle Ergänzungen werden gebraucht, manche aber mehrfach.

```
1 #OPTIONAL
2 #IMPLIED
3 (#PCDATA)
4 (PCDATA)
5 #PCDATA
6 EMPTY
7 #EMPTY
```

```
<!-- DTD fuer den Pegel -->
<!ELEMENT Pegel (Stammdaten, Wasserstand)>
<!ATTLIST Pegel  Name CDATA #REQUIRED
                 Mst_Nr CDATA #REQUIRED
                 Flussgebiet CDATA __2__>

<!ELEMENT Stammdaten (Einzugsgebiet,
       Entfernung_Muendung, Betreiber,
       Hochwassermeldepegel, Meldestufe1,
       Meldestufe2, Meldestufe3, HHW)>

<!ELEMENT Einzugsgebiet ___3___>
<!ATTLIST Einzugsgebiet Einheit CDATA #REQUIRED>

<!ELEMENT Entfernung_Muendung ___3__>
<!ATTLIST Entfernung_Muendung Einheit CDATA #REQUIRED>

<!ELEMENT Betreiber ___3__>

<!ELEMENT Hochwassermeldepegel ___3__>

<!ELEMENT Meldestufe1 ___3__>
<!ATTLIST Meldestufe1 Einheit CDATA #REQUIRED>
...
<!ELEMENT HHW ___3__>
<!ATTLIST HHW  wann CDATA #REQUIRED
               Einheit CDATA #REQUIRED>

<!ELEMENT Wasserstand (Wert)>
<!ATTLIST Wasserstand Einheit CDATA #REQUIRED>

<!ELEMENT Wert _____6____>
<!ATTLIST Wert W CDATA #REQUIRED
               T CDATA #REQUIRED
               U CDATA #REQUIRED>
```

Aufgabe 3:

Nehmen wir an, das Element *Hochwassermeldepegel* wäre jetzt ein leeres Element mit dem Attribut *A*, das nur die Werte *ja, nein, k.A.* annehmen darf, wobei *k.A.* der Default-Wert ist. Ergänzen Sie für diese neue Vorschrift die Lücken in der DTD-Angabe unten.

```
<!ELEMENT Hochwassermeldepegel __EMPTY__>

<!ATTLIST Hochwassermeldepegel A ___ ( ja | nein | k.A. ) "k.A." ___>
```

Aufgabe 4:

Ein XML-Schema zum Pegel-Dokument sieht wie folgt aus. Füllen Sie die Lücken. Orientieren Sie sich bei den gewünschten Angaben an den Vorgaben der DTD aus Aufgabe 2. Zusätzlich bzw. abweichend soll gelten:

- Das Attribut *W* im Element *Wert* ist eine positive ganze Zahl. Die Attribute *T* und *U* seien als Zeichenketten deklariert.
- Die Inhalte der Elemente *Einzugsgebiet, Entfernung_Muendung,* die *Meldestufen* und *HHW* seien als Dezimalzahlen deklariert.
- Alle Attribute *Einheit* seien vom Typ Zeichenkette.
- Beachten Sie, dass das Attribut *Flussgebiet* weiterhin optional ist.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Pegel" type="TPegel"/>
  <xsd:complexType name="TPegel">
    <__xsd:sequence__>
      <xsd:element name="Stammdaten" type="__TStammdaten__"/>
      <xsd:element name="Wasserstand" type="TWasserstand"/>
    </__xsd:sequence__>
    <xsd:attribute name="Name" type="xsd:string" use="required"/>
    <xsd:attribute name="Mst_Nr" type="xsd:integer" use="required"/>
    <xsd:attribute name="Flussgebiet" type="xsd:string"
          use="__optional__"/>
  </xsd:complexType>

  <xsd:complexType name="__TStammdaten__">
    <xsd:sequence>
      <xsd:element name="Einzugsgebiet" type="__TEinheit__"/>
      <xsd:element name="Entfernung_Muendung" type="__TEinheit__"/>
      <xsd:element name="Betreiber" type="xsd:string"/>
      <xsd:element name="Hochwassermeldepegel" type="xsd:string"/>
      <xsd:element name="Meldestufe1" type="TEinheit"/>
```

beliebig gewählt, aber übereinstimmend

```
      <xsd:element name="Meldestufe2" type="TEinheit"/>
      <xsd:element name="Meldestufe3" type="TEinheit"/>
      <xsd:element name="HHW" type="__TEinheitWann__"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="TEinheit">
    <xsd:simpleContent>
      <xsd:extension base="xsd:decimal">
        <xsd:attribute name="Einheit" type="xsd:string"
              use="required"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>

  <xsd:complexType name="TEinheitWann">
    <xsd:simpleContent>
      <xsd:extension base="___TEinheit___">
        <xsd:attribute name="wann" type="xsd:string"
              use="required"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>

  <xsd:complexType name="TWasserstand">
    <xsd:sequence>
      <xsd:element name="Wert">
        <xsd:complexType>
          <xsd:attribute name="W" type="__xsd:positiveInteger___"
                use="required"/>
          <xsd:attribute name="T" type="xsd:string" use="required"/>
          <xsd:attribute name="U" type="xsd:string" use="required"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="___Einheit__" type="xsd:string"
          use="required"/>
  </xsd:complexType>
</xsd:schema>
```

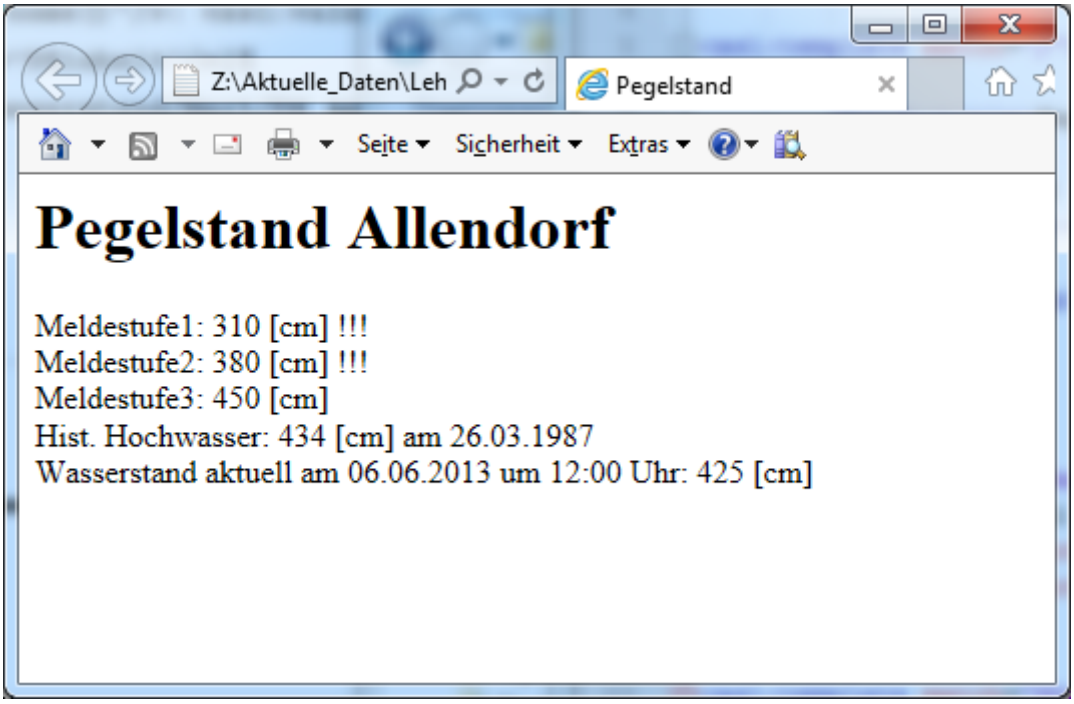
Aufgabe 5:

Ergänzen Sie die Lücken im Stylesheet, damit unser, ggf. korrigiertes Pegel-Dokument aus Aufgabe 1 die unten gezeigte Ausgabe erzeugt. Übertrifft ein aktueller Pegelstand eine oder mehrere Meldestufen, sollen drei Ausrufezeichen hinter dem jeweiligen Meldestufenwert erscheinen.

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet version='1.0'
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
    <head><title>Pegelstand</title></head>
    <body>
      <h1>Pegelstand
        <xsl:value-of select="__Pegel/@Name__"/>
      </h1>
      <xsl:apply-templates select="Pegel/*"/>
    </body>
  </html>
</xsl:template>

<xsl:template match="____Stammdaten____">
  <xsl:_apply-templates__ select="Meldestufe1|Meldestufe2|Meldestufe3"/>
  Hist. Hochwasser: <xsl:value-of select="HHW"/>
  [<xsl:value-of select="__HHW/@Einheit__"/>]
  am <xsl:value-of select="__HHW/@wann__"/><br/>
</xsl:template>

<xsl:template match="Meldestufe1|Meldestufe2|Meldestufe3">
  <xsl:value-of select="___name()___"/>: <xsl:value-of select="."/>
  [<xsl:value-of select="@Einheit"/>]
  <xsl:if test="../../Wasserstand/Wert/@W &gt; __ . ___"> !!!</xsl:if>
  <br/>
</xsl:template>

<xsl:template match="___Wasserstand___">
  Wasserstand aktuell am <xsl:value-of select="Wert/@T"/> um
  <xsl:value-of select="Wert/@U"/> Uhr:
  <xsl:value-of select="Wert/@W"/> [<xsl:value-of select="@Einheit"/>]
</xsl:template>
</xsl:stylesheet>
```

auch: text()

Ausgabe:



Aufgabe 6:

Füllen Sie die Lücken in der folgenden XQuery auf dem ggf. korrigierten Dokument **`Pegel.xml`**, sodass das unten gezeigte Ergebnis (also der Pegelstand und die höchste überschrittene Meldestufe) geliefert wird?

```
let $p := fn:doc("Pegel.xml")/Pegel
let $m := ($p/Stammdaten/Meldestufe3[$p/Wasserstand/Wert/@W > .],
           $p/Stammdaten/Meldestufe2[$p/Wasserstand/Wert/@W > .],
           $p/Stammdaten/Meldestufe1[$p/Wasserstand/Wert/@W > .],
           <Meldestufen_nicht/>)
let $x := $m[1]
return
    ___<Pegelmeldung Name="{$p/@Name}">
    _____<Pegelstand W="{$p/Wasserstand/Wert/@W}"/>
    _____<Warnung>{fn:name($x)} ueberschritten</Warnung>
    ___</Pegelmeldung>
```

Ausgabe:

```
<Pegelmeldung Name="Allendorf">
    <Pegelstand W="425"/>
    <Warnung>Meldestufe2 ueberschritten</Warnung>
</Pegelmeldung>
```

Aufgabe 7:

Gegeben sei die folgende Datenbanktabelle mit Namen **PEGELSTAENDE**.

GEWAESSER	STATION	WASSER-STAND
Werra	Allendorf	139
Werra	Heldra	203
Weser	Bad Karlshafen	160
Lahn	Marburg	173
Fulda	Bad Hersfeld	237
Fulda	Rotenburg	165

Füllen Sie die Lücken in der SQL/XML-Abfrage auf der PEGELSTAENDE-Tabelle, die als Ergebnis die unten gezeigte Ausgabe liefert.

```
SELECT XMLELEMENT (NAME "Flussgebiet",
           XMLATTRIBUTES (GEWAESSER AS "Fluss"),
           XMLAGG (
             XMLELEMENT (NAME "Station",
               XMLATTRIBUTES (STATION AS "Name"),
               WASSERSTAND
             )
           )
       )
FROM PEGELSTAENDE
GROUP BY GEWAESSER;
```

Ausgabe:

```
<Flussgebiet Fluss="Werra">
  <Station Name="Allendorf">139</Station>
  <Station Name="Heldra">203</Station>
</Flussgebiet>
<Flussgebiet Fluss="Weser">
  <Station Name="Bad Karlshafen">160</Station>
</Flussgebiet>
...
```

Aufgabe 8:

In unserer Sammlung animierter Verkehrszeichen betrachten wir heute das Gefahrenzeichen 129 *Ufer*, das allerdings zum 01.04.2013 aus dem Regelkatalog gestrichen wurde.

Ergänzen Sie die Lücken durch Angabe der passenden Ziffer! Hinweis: Nicht alle Ergänzungen werden gebraucht, manche aber mehrfach.

```
1 circle
2 drive
3 fill
4 line
5 path
6 polygon
7 wave
```


```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg">
<rect x="0" y="0" width="600" height="600" fill="white"/>
<__4__ x1="50" y1="500" x2="550" y2="500"
  style="stroke:red; stroke-width:50px; stroke-linecap:round" />
<__4__ x1="50" y1="500" x2="300" y2="67"
  style="stroke:red; stroke-width:50px; stroke-linecap:round" />
<__4__ x1="300" y1="67" x2="550" y2="500"
  style="stroke:red; stroke-width:50px; stroke-linecap:round" />
<__5__ style="stroke:black; stroke-width:2px" fill="none"
  d="M 260 50 A 43 43 60 0 1 340 50 L 590 480
    A 43 43 -60 0 1 550 545 L 50 545
    A 43 43 -60 0 1 10 480 L 260 50" />

<__6__ points="110,465 168,368 250,368 250,465"
  style="fill:black;stroke:none"/>

<g id="car">
  <__6__ points="160,345 160,315 200,280 280,280 310,310 360,320 360,345"
    style="fill:black;stroke:none"/>
  <__6__ id="fensterhinten" points="190,310 210,285 238,285 238,310"
    style="fill:white;stroke:none"/>
  <__6__ id="fenstervorne" points="245,310 245,285 275,285 300,310"
    style="fill:white;stroke:none"/>
```

```
  <__1__ id="wheel1" cx="200" cy="345" r="20"
    style="fill:black; stroke:white; stroke-width:4px"/>
  <__1__ id="wheel2" cx="320" cy="345" r="20"
    style="fill:black; stroke:white; stroke-width:4px"/>
  <animateTransform attributeName="transform" type="rotate"
    from="6 220 600" to="22 230 600" begin="0s" dur="5s"
    repeatCount="indefinite"/>
</g>

<__5__ style="stroke:black; stroke-width:15px" fill="none"
  d="M 250,420 q 25,-15 50,0 t 50,0" />
<__5__ style="stroke:black; stroke-width:15px" fill="none"
  d="M 350,420 q 25,-15 50,0 t 50,0" />
<__5__ style="stroke:black; stroke-width:15px" fill="none"
  d="M 250,450 q 25,-15 50,0 t 50,0" />
<__5__ style="stroke:black; stroke-width:15px" fill="none"
  d="M 350,450 q 25,-15 50,0 t 50,0" />
<__5__ style="stroke:black; stroke-width:15px" fill="none"
  d="M 449,451 q 15,-6 17, -7" />
<line style="stroke:white; stroke-width:10px" x1="255" y1="380" x2="255" y2="460"/>
</svg>
```

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

Nachname: Vorname:

Matr.Nr.: Studiengang:

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit beträgt 120 Minuten.

Aufgabe	Punkte max.	Punkte erreicht
1	6	
2	3	
3	6	
4	5	
5	4	
6	4	
7	6	
Summe	**34**	

Am 2. März ist es wieder so weit: die Oscarverleihung in Los Angeles steht an. Hierzu betrachten wir einen Ausschnitt aus der Liste der Nominierungen im Dokument **`Oscar.xml`**.

```
<?xml version="1.0"?>
<Oscar Datum="2014-03-02" Ort="Dolby Theatre Los Angeles">
  <Filme>
    <Titel id="f17">American Hustle</Titel>
    <Titel id="f18">August: Osage County</Titel>
    <Titel id="f19">The Wolf of Wall Street</Titel>
    <Titel id="f20">12 Years a Slave</Titel>
  </Filme>
  <Nominierungen>
    <BesterFilm>
      <Film FilmId="f17" Regie="David O. Russell"/>
      <Film FilmId="f20" Regie="Steve McQueen"/>
    </BesterFilm>
    <BesteRegie>
      <Name FilmId="f17">David O. Russell</Name>
      <Name FilmId="f19">Martin Scorsese</Name>
    </BesteRegie>
    <BesterHauptdarsteller>
      <Name FilmId="f19">Leonardo DiCaprio</Name>
      <Name FilmId="f17">Christian Bale</Name>
    </BesterHauptdarsteller>
    <BesteHauptdarstellerin>
      <Name FilmId="f18">Meryl Streep</Name>
    </BesteHauptdarstellerin>
  </Nominierungen>
</Oscar>
```

Aufgabe 1:

Gegeben ist die unten stehende DTD für das Oscar-Dokument oben. Füllen Sie die Lücken! Es gelten die folgenden Vorgaben:

- Die Elemente *Titel* und *Name* erlauben nur Textinhalt, keine Unterelemente.
- Die Attribute *id* und *FilmId* sind eindeutige Bezeichner für Elemente bzw. einzelne Verweise darauf.
- Das Attribut *Ort* kann weggelassen werden.
- Das Element *Film* muss leer sein.
- Das Element *Filme* enthält einen oder mehr *Titel*.

```
<!ELEMENT Oscar (Filme, Nominierungen)>

<!ATTLIST Oscar Datum CDATA #REQUIRED

                Ort CDATA _______________>

<!ELEMENT Filme (____________)>

<!ELEMENT Titel (_____________)>

<!ATTLIST Titel id _________ #REQUIRED>

<!ELEMENT ___________________ (BesterFilm | BesteRegie |

        BesterHauptdarsteller | BesteHauptdarstellerin)*>

<!ELEMENT BesterFilm (Film+)>

<!ELEMENT Film __________>

<!ATTLIST Film FilmId _________ #REQUIRED

               Regie CDATA #REQUIRED>

<!ELEMENT BesteRegie (Name+)>

<!ELEMENT Name (_____________)>

<!ATTLIST Name FilmId _________ #REQUIRED>

<!ELEMENT BesterHauptdarsteller (Name+)>

<!ELEMENT BesteHauptdarstellerin (Name+)>
```

Aufgabe 2:

Kreuzen Sie die richtigen Aussagen an!

( ) Im XML-Dokument oben hätte man statt `<BesterFilm>` auch `<Bester Film>` schreiben können.

( ) Weil in der DTD das Attribut *Datum* vor dem Attribut *Ort* vereinbart wird, können die Attribute im XML-Dokument auch nur in dieser Reihenfolge stehen.

( ) Ob ein Dokument wohlgeformt ist, kann nur mit einer DTD oder einem XML-Schema geprüft werden.

( ) Die Schreibweisen `<beispiel/>` und `<beispiel></beispiel>` sind für leere Elemente gleichwertig.

( ) Elemente, die Textinhalt besitzen, dürfen nie zusätzlich Unterelemente aufweisen.

( ) Wäre der Film *Tim & Struppi* nominiert, sind die beiden Schreibweisen
`<Titel id="f99">Tim & Struppi</Titel>` und
`<Titel id="f99">Tim &amp; Struppi</Titel>` erlaubt.

Aufgabe 3:

Ein XML-Schema zum Oscar-Dokument sieht wie folgt aus. Füllen Sie die Lücken! Orientieren Sie sich bei den gewünschten Angaben an den Vorgaben zur DTD aus Aufgabe 1. Zusätzlich bzw. abweichend soll gelten:

- Das Attribut *Datum* hat einen geeigneten Datumstyp.
- Für das Element *Nominierungen* soll gelten, dass die Unterelemente in beliebiger Reihenfolge auftreten können, jedes Unterelement jedoch genau einmal.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<xsd:element name="Oscar" type="OscarType"/>

<xsd:complexType name="_________________">
  <xsd:sequence>
    <xsd:element name="Filme" type="FilmeType"/>
    <xsd:element name="Nominierungen" type="NominierungenType"/>
  </xsd:sequence>
  <xsd:attribute name="Datum" type="__________________" use="required"/>
  <xsd:attribute name="Ort" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="FilmeType">
  <xsd:sequence>
    <xsd:element name="Titel" type="TitelType" maxOccurs="_________________"/>
  </xsd:sequence>
</xsd:complexType>
```

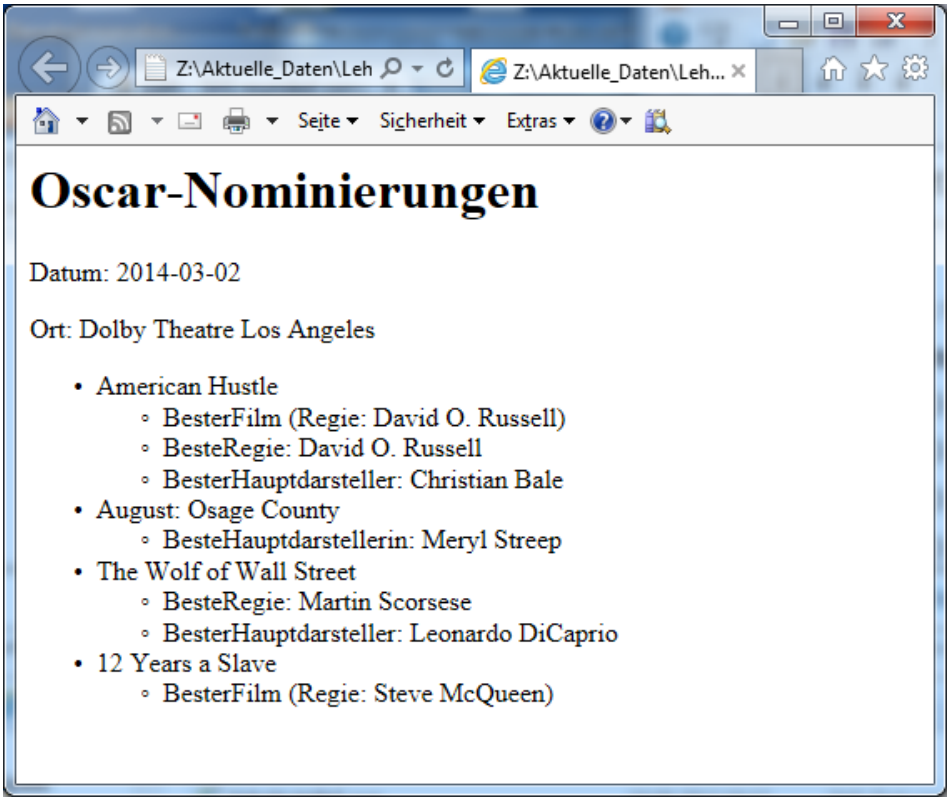
```
<xsd:complexType name="TitelType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="id" type="____________" use="required"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="NominierungenType">
  <xsd:_______>
    <xsd:element name="BesteRegie" type="BestePersonType"/>
    <xsd:element name="BesterHauptdarsteller" type="BestePersonType"/>
    <xsd:element name="BesteHauptdarstellerin" type="BestePersonType"/>
    <xsd:element name="BesterFilm" type="BesterFilmType"/>
  </xsd:_______>
</xsd:complexType>

<xsd:complexType name="BesterFilmType">
  <xsd:sequence>
    <xsd:element name="Film" maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:attribute name="_________" type="xsd:IDREF" use="required"/>
        <xsd:attribute name="_________" type="xsd:string" use="required"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="_____________________________">
  <xsd:sequence>
    <xsd:element name="Name" maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:____________________>
          <xsd:extension base="xsd:string">
            <xsd:attribute name="FilmId" type="xsd:IDREF" use="required"/>
          </xsd:extension>
        </xsd:____________________>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>

</xsd:schema>
```

Aufgabe 4:

Ergänzen Sie die Lücken im Stylesheet, damit unser Oscar-Dokument die unten gezeigte Ausgabe erzeugt.

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

<xsl:template match="/">
  <html><body>
    <h1>Oscar-Nominierungen</h1>
    <p>Datum: <xsl:value-of select="____________________"/></p>
    <p>Ort: <xsl:value-of select="____________________"/></p>
    <ul><xsl:_____________________ select="Oscar/Filme/Titel"/></ul>
  </body></html>
</xsl:template>

<xsl:template match="Titel">
  <li>
    <xsl:value-of select="_____________"/>
    <ul>
      <xsl:apply-templates
        select="/Oscar/Nominierungen/*/*[_________ = current()/@id]"/>
    </ul>
  </li>
</xsl:template>

<xsl:template match="_____________">
  <li>
    <xsl:value-of select="name(..)"/> (Regie:
        <xsl:value-of select="@Regie"/>)
  </li>
</xsl:template>

<xsl:template match="_____________">
  <li>
    <xsl:value-of select="_____________"/>: <xsl:value-of select="."/>
  </li>
</xsl:template>

</xsl:stylesheet>
```

Ausgabe:



Aufgabe 5:

Geben Sie die Ausgabe der folgenden XQuery auf dem Dokument **`Oscar.xml`** an!

```
<Filmliste>{
  let $o := fn:doc("Oscar.xml")/Oscar
  for $titel in $o/Filme/Titel
  where every $fid in $o/Nominierungen/BesterFilm/Film/@FilmId
        satisfies $fid != $titel/@id
  return
    <Film>{ fn:string($titel) }</Film>
}</Filmliste>
```

Ergebnis:

Aufgabe 6:

Gegeben seien die beiden Tabellen **FILME** und **NOMINIERUNGEN.**

FILME

FID	TITEL	REGISSEUR
17	American Hustle	David O. Russell
18	August: Osage County	John Wells
19	The Wolf of Wall Street	M. Scorsese
20	12 Years a Slave	Steve McQueen

NOMINIERUNGEN

KATEGORIE	NAME	INFILM
Bester Film	*NULL*	17
Bester Film	*NULL*	20
Regie	David O. Russell	17
Regie	M. Scorsese	19
Hauptdarsteller	Christian Bale	17
Hauptdarsteller	Leonardo DiCaprio	19
Hauptdarstellerin	Meryl Streep	18

Füllen Sie die Lücken in der SQL/XML-Abfrage auf den Tabellen oben, die als Ergebnis die unten gezeigte Ausgabe liefert. Das XML-Attribut **AnzNom** enthält die Anzahl der Nominierungen des jeweiligen Films in unserer Beispieltabelle.

```
SELECT ____________________ (_____________________________,
        ____________________ (_____________________________________,
          (SELECT COUNT(*)
           FROM NOMINIERUNGEN
           WHERE FILME.FID = NOMINIERUNGEN.INFILM) AS "AnzNom"
        ),
        ___________________
      )
FROM FILME;
```

Ausgabe:

```
<Film Regie="David O. Russell" AnzNom="3">American Hustle</Film>
<Film Regie="John Wells" AnzNom="1">August: Osage County</Film>
<Film Regie="M. Scorsese" AnzNom="2">The Wolf of Wall Street</Film>
<Film Regie="Steve McQueen" AnzNom="1">12 Years a Slave</Film>
```

Aufgabe 7:

In unserer Sammlung animierter Verkehrszeichen betrachten wir das Gefahrenzeichen „Radfahrer kreuzen (Aufstellung rechts)“ - StVO Verkehrszeichen-Nr. 138-10. Ergänzen Sie die Lücken!

```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg"
   xmlns:xlink="http://www.w3.org/1999/xlink">

<rect x="0" y="0" width="600" height="600" _____="white"/>

<__________>
   <g id="bike">
      <polygon points="95,-90 110,-85 140,-85
         140,-95 95,-95" fill="black" stroke="none"/>
      <g fill="none" stroke="black" stroke-width="10px"
            stroke-linecap="round">
         <path d="M 26 -70 Q 32 -100 47 -100 L 57 -100"/>
         <line x1="3" y1="0" x2="26" y2="-70"/>
         <line x1="26" y1="-70" x2="90" y2="0"/>
         <line x1="120" y1="-70" x2="155" y2="0"/>
         <line x1="90" y1="0" x2="120" y2="-85"/>
         <line x1="26" y1="-70" x2="120" y2="-70"/>
         <line x1="90" y1="0" x2="155" y2="0"/>
         <circle cx="155" cy="0" r="40"/>
         <circle cx="3" cy="0" r="40"/>
      </g>
   </g>
</__________>

<use __________="400" xlink:href="_____________">
   <animate attributeName="________" from="550" to="-80"
      dur="6s" repeatCount="indefinite"/>
</use>

<g fill="_________________" stroke="none">
   <polygon points="550,495 420,270 800,270 800,495"/>
   <polygon points="0,495 50,495 180,270 0,270"/>
</g>

<polygon points="50,500 550,500 300,67" fill="_________________"
      stroke="red" stroke-width="50px" stroke-linejoin="round"/>

<path fill="none" __________="black" stroke-width="2px"
   d="M 260 50 A 43 43 60 0 1 340 50 L 590 480 A 43 43 -60 0 1 550 545 L 50 545
      A 43 43 -60 0 1 10 480 L 260 50"/>

</svg>
```

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

Nachname: Vorname:

Matr.Nr.: Studiengang:

MUSTERLÖSUNG

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit beträgt 120 Minuten.

Aufgabe	Punkte max.	Punkte erreicht
1	6	
2	3	
3	6	
4	5	
5	4	
6	4	
7	6	
Summe	**34**	

Am 2. März ist es wieder so weit: die Oscarverleihung in Los Angeles steht an. Hierzu betrachten wir einen Ausschnitt aus der Liste der Nominierungen im Dokument **`Oscar.xml`**.

```
<?xml version="1.0"?>
<Oscar Datum="2014-03-02" Ort="Dolby Theatre Los Angeles">
  <Filme>
    <Titel id="f17">American Hustle</Titel>
    <Titel id="f18">August: Osage County</Titel>
    <Titel id="f19">The Wolf of Wall Street</Titel>
    <Titel id="f20">12 Years a Slave</Titel>
  </Filme>
  <Nominierungen>
    <BesterFilm>
      <Film FilmId="f17" Regie="David O. Russell"/>
      <Film FilmId="f20" Regie="Steve McQueen"/>
    </BesterFilm>
    <BesteRegie>
      <Name FilmId="f17">David O. Russell</Name>
      <Name FilmId="f19">Martin Scorsese</Name>
    </BesteRegie>
    <BesterHauptdarsteller>
      <Name FilmId="f19">Leonardo DiCaprio</Name>
      <Name FilmId="f17">Christian Bale</Name>
    </BesterHauptdarsteller>
    <BesteHauptdarstellerin>
      <Name FilmId="f18">Meryl Streep</Name>
    </BesteHauptdarstellerin>
  </Nominierungen>
</Oscar>
```

Aufgabe 1:

Gegeben ist die unten stehende DTD für das Oscar-Dokument oben. Füllen Sie die Lücken! Es gelten die folgenden Vorgaben:

- Die Elemente *Titel* und *Name* erlauben nur Textinhalt, keine Unterelemente.
- Die Attribute *id* und *FilmId* sind eindeutige Bezeichner für Elemente bzw. einzelne Verweise darauf.
- Das Attribut *Ort* kann weggelassen werden.
- Das Element *Film* muss leer sein.
- Das Element *Filme* enthält einen oder mehr *Titel*.

```
<!ELEMENT Oscar (Filme, Nominierungen)>

<!ATTLIST Oscar Datum CDATA #REQUIRED

                Ort CDATA _#IMPLIED_>

<!ELEMENT Filme (_Titel+_)>

<!ELEMENT Titel (_#PCDATA_)>

<!ATTLIST Titel id __ID__ #REQUIRED>

<!ELEMENT _Nominierungen_ (BesterFilm | BesteRegie |

        BesterHauptdarsteller | BesteHauptdarstellerin)*>

<!ELEMENT BesterFilm (Film+)>

<!ELEMENT Film _EMPTY_>

<!ATTLIST Film FilmId _IDREF_ #REQUIRED

               Regie CDATA #REQUIRED>

<!ELEMENT BesteRegie (Name+)>

<!ELEMENT Name (_#PCDATA_)>

<!ATTLIST Name FilmId _IDREF_ #REQUIRED>

<!ELEMENT BesterHauptdarsteller (Name+)>

<!ELEMENT BesteHauptdarstellerin (Name+)>
```

Aufgabe 2:

Kreuzen Sie die richtigen Aussagen an!

( ) Im XML-Dokument oben hätte man statt `<BesterFilm>` auch `<Bester Film>` schreiben können.

( ) Weil in der DTD das Attribut *Datum* vor dem Attribut *Ort* vereinbart wird, können die Attribute im XML-Dokument auch nur in dieser Reihenfolge stehen.

( ) Ob ein Dokument wohlgeformt ist, kann nur mit einer DTD oder einem XML-Schema geprüft werden.

(X) Die Schreibweisen `<beispiel/>` und `<beispiel></beispiel>` sind für leere Elemente gleichwertig.

( ) Elemente, die Textinhalt besitzen, dürfen nie zusätzlich Unterelemente aufweisen.

( ) Wäre der Film *Tim & Struppi* nominiert, sind die beiden Schreibweisen
`<Titel id="f99">Tim & Struppi</Titel>` und
`<Titel id="f99">Tim &amp; Struppi</Titel>` erlaubt.

Aufgabe 3:

Ein XML-Schema zum Oscar-Dokument sieht wie folgt aus. Füllen Sie die Lücken. Orientieren Sie sich bei den gewünschten Angaben an den Vorgaben zur DTD aus Aufgabe 1. Zusätzlich bzw. abweichend soll gelten:

- Das Attribut *Datum* hat einen geeigneten Datumstyp.
- Für das Element *Nominierungen* soll gelten, dass die Unterelemente in beliebiger Reihenfolge auftreten können, jedes Unterelement jedoch genau einmal.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<xsd:element name="Oscar" type="OscarType"/>

<xsd:complexType name="_OscarType_">
  <xsd:sequence>
    <xsd:element name="Filme" type="FilmeType"/>
    <xsd:element name="Nominierungen" type="NominierungenType"/>
  </xsd:sequence>
  <xsd:attribute name="Datum" type="_xsd:date_" use="required"/>
  <xsd:attribute name="Ort" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="FilmeType">
  <xsd:sequence>
    <xsd:element name="Titel" type="TitelType" maxOccurs="__unbounded__"/>
  </xsd:sequence>
</xsd:complexType>
```

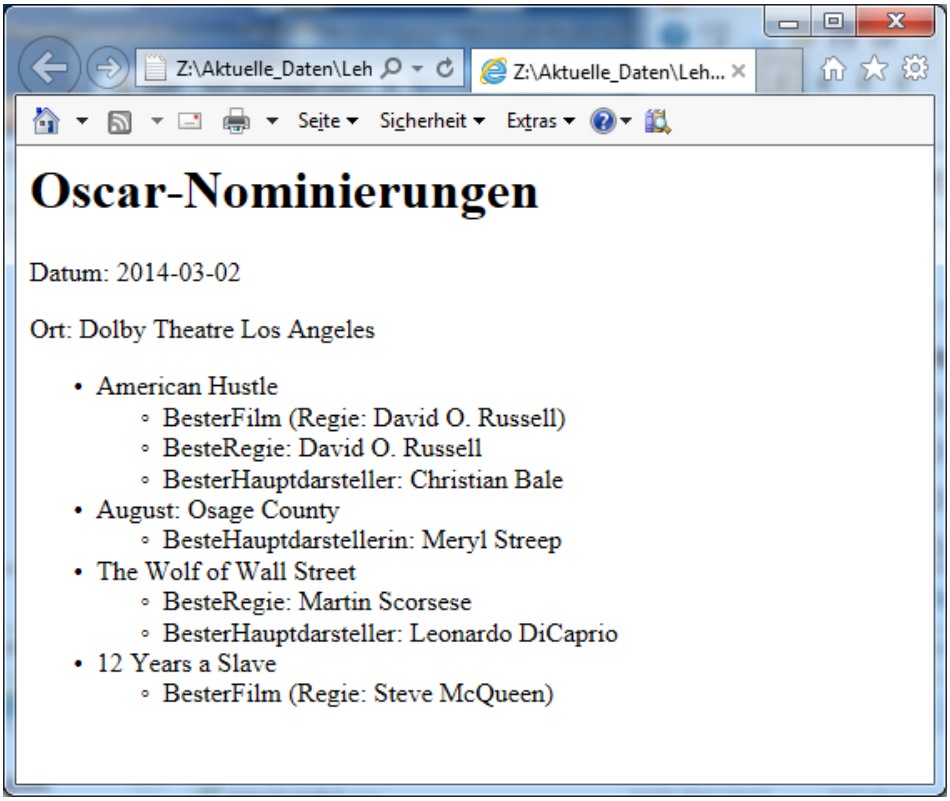
```
<xsd:complexType name="TitelType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="id" type="_xsd:ID_" use="required"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="NominierungenType">
  <xsd:_all_>
    <xsd:element name="BesteRegie" type="BestePersonType"/>
    <xsd:element name="BesterHauptdarsteller" type="BestePersonType"/>
    <xsd:element name="BesteHauptdarstellerin" type="BestePersonType"/>
    <xsd:element name="BesterFilm" type="BesterFilmType"/>
  </xsd:_all_>
</xsd:complexType>

<xsd:complexType name="BesterFilmType">
  <xsd:sequence>
    <xsd:element name="Film" maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:attribute name="_FilmId_" type="xsd:IDREF" use="required"/>
        <xsd:attribute name="_Regie_" type="xsd:string" use="required"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="__BestePersonType__">
  <xsd:sequence>
    <xsd:element name="Name" maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:_simpleContent_>
          <xsd:extension base="xsd:string">
            <xsd:attribute name="FilmId" type="xsd:IDREF" use="required"/>
          </xsd:extension>
        </xsd:_simpleContent_>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>

</xsd:schema>
```

Aufgabe 4:

Ergänzen Sie die Lücken im Stylesheet, damit unser Oscar-Dokument die unten gezeigte Ausgabe erzeugt.

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

<xsl:template match="/">
  <html><body>
    <h1>Oscar-Nominierungen</h1>
    <p>Datum: <xsl:value-of select="__Oscar/@Datum__"/></p>
    <p>Ort: <xsl:value-of select="__Oscar/@Ort__"/></p>
    <ul><xsl:_apply-templates_ select="Oscar/Filme/Titel"/></ul>
  </body></html>
</xsl:template>

<xsl:template match="Titel">
  <li>
    <xsl:value-of select="_._"/>
    <ul>
      <xsl:apply-templates
        select="/Oscar/Nominierungen/*/*[_@FilmId_ = current()/@id]"/>
    </ul>
  </li>
</xsl:template>

<xsl:template match="__Film__">
  <li>
    <xsl:value-of select="name(..)"/> (Regie:
        <xsl:value-of select="@Regie"/>)
  </li>
</xsl:template>

<xsl:template match="__Name__">
  <li>
    <xsl:value-of select="_name(..)_"/>: <xsl:value-of select="."/>
  </li>
</xsl:template>

</xsl:stylesheet>
```

Ausgabe:



Aufgabe 5:

Geben Sie die Ausgabe der folgenden XQuery auf dem Dokument **`Oscar.xml`** an!

```
<Filmliste>{
  let $o := fn:doc("Oscar.xml")/Oscar
  for $titel in $o/Filme/Titel
  where every $fid in $o/Nominierungen/BesterFilm/Film/@FilmId
        satisfies $fid != $titel/@id
  return
    <Film>{ fn:string($titel) }</Film>
}</Filmliste>
```

Ergebnis:

```
<Filmliste>
    <Film>August: Osage County</Film>
    <Film>The Wolf of Wall Street</Film>
</Filmliste>
```

Aus der Liste der Filme (Titel) diejenigen, die in unserem Dokument nicht in der Kategorie „Bester Film" nominiert wurden.

Aufgabe 6:

Gegeben seien die beiden Tabellen **FILME** und **NOMINIERUNGEN.**

FILME

FID	TITEL	REGISSEUR
17	American Hustle	David O. Russell
18	August: Osage County	John Wells
19	The Wolf of Wall Street	M. Scorsese
20	12 Years a Slave	Steve McQueen

NOMINIERUNGEN

KATEGORIE	NAME	INFILM
Bester Film	*NULL*	17
Bester Film	*NULL*	20
Regie	David O. Russell	17
Regie	M. Scorsese	19
Hauptdarsteller	Christian Bale	17
Hauptdarsteller	Leonardo DiCaprio	19
Hauptdarstellerin	Meryl Streep	18

Füllen Sie die Lücken in der SQL/XML-Abfrage auf den Tabellen oben, die als Ergebnis die unten gezeigte Ausgabe liefert. Das XML-Attribut **AnzNom** enthält die Anzahl der Nominierungen des jeweiligen Films in unserer Beispieltabelle.

```
SELECT _XMLELEMENT_ (_NAME "Film"_,
         __XMLATTRIBUTES__ (__REGISSEUR AS "Regie"__,
           (SELECT COUNT(*)
            FROM NOMINIERUNGEN
            WHERE FILME.FID = NOMINIERUNGEN.INFILM) AS "AnzNom"
         ),
         __TITEL__
       )
FROM FILME;
```

Ausgabe:

```
<Film Regie="David O. Russell" AnzNom="3">American Hustle</Film>
<Film Regie="John Wells" AnzNom="1">August: Osage County</Film>
<Film Regie="M. Scorsese" AnzNom="2">The Wolf of Wall Street</Film>
<Film Regie="Steve McQueen" AnzNom="1">12 Years a Slave</Film>
```

Aufgabe 7:

In unserer Sammlung animierter Verkehrszeichen betrachten wir das Gefahrenzeichen „Radfahrer kreuzen (Aufstellung rechts)“ - StVO Verkehrszeichen-Nr. 138-10. Ergänzen Sie die Lücken!

```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg"
   xmlns:xlink="http://www.w3.org/1999/xlink">

<rect x="0" y="0" width="600" height="600" _fill_="white"/>

<__defs__>
   <g id="bike">
      <polygon points="95,-90 110,-85 140,-85
         140,-95 95,-95" fill="black" stroke="none"/>
      <g fill="none" stroke="black" stroke-width="10px"
            stroke-linecap="round">
         <path d="M 26 -70 Q 32 -100 47 -100 L 57 -100"/>
         <line x1="3" y1="0" x2="26" y2="-70"/>
         <line x1="26" y1="-70" x2="90" y2="0"/>
         <line x1="120" y1="-70" x2="155" y2="0"/>
         <line x1="90" y1="0" x2="120" y2="-85"/>
         <line x1="26" y1="-70" x2="120" y2="-70"/>
         <line x1="90" y1="0" x2="155" y2="0"/>
         <circle cx="155" cy="0" r="40"/>
         <circle cx="3" cy="0" r="40"/>
      </g>
   </g>
</__defs__>

<use __y___="400" xlink:href="__#bike__">
   <animate attributeName="_x_" from="550" to="-80"
      dur="6s" repeatCount="indefinite"/>
</use>

<g fill="__white__" stroke="none">
   <polygon points="550,495 420,270 800,270 800,495"/>
   <polygon points="0,495 50,495 180,270 0,270"/>
</g>

<polygon points="50,500 550,500 300,67" fill="__none__"
      stroke="red" stroke-width="50px" stroke-linejoin="round"/>

<path fill="none" _stroke_="black" stroke-width="2px"
   d="M 260 50 A 43 43 60 0 1 340 50 L 590 480 A 43 43 -60 0 1 550 545 L 50 545
      A 43 43 -60 0 1 10 480 L 260 50"/>

</svg>
```

Hier weiße Abdeckflächen, damit das Rad verschwindet

Hier leer, damit das Rad durchscheint

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

Nachname: Vorname:

Matr.Nr.: Studiengang:

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 90 Minuten.

Aufgabe	**Punkte max.**	**Punkte erreicht**
1	1 + 2	
2	4	
3	2	
4	6	
5	6	
6	4	
7	4	
8	4 + 1	
Summe	**34**	

Aufgabe 1:

Wir stehen immer noch unter dem Eindruck der Fußballweltmeisterschaft in Brasilien. Hier ist der Spielplan ab dem Viertelfinale (verkürzt).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="SpielplanStyle.xsl"?>
<!DOCTYPE Spielplan SYSTEM "Spielplan.dtd">
<Spielplan Turnier="Fußball-WM 2014 Brasilien">
<Runde Bezeichnung="Viertelfinale">
  <Spiel ID="VF1" Tag="04.07.2014" Zeit="18:00" Spielort="Rio d. J.">
    <Mannschaft>Frankreich</Mannschaft>
    <Mannschaft>Deutschland</Mannschaft>
    <Ergebnis ToreM1="0" ToreM2="1"/>
  </Spiel>
  <Spiel ID="VF2" Tag="04.07.2014" Zeit="22:00" Spielort="Fortaleza">
    <Mannschaft>Brasilien</Mannschaft>
    <Mannschaft>Kolumbien</Mannschaft>
    <Ergebnis ToreM1="2" ToreM2="1"/>
  </Spiel>
  <Spiel ID="VF3" Tag="05.07.2014" Zeit="18:00" Spielort="Brasilia">
    <Mannschaft>Argentinien</Mannschaft>
    <Mannschaft>Belgien</Mannschaft>
    <Ergebnis ToreM1="1" ToreM2="0"/>
  </Spiel>
  <Spiel ID="VF4" Tag="05.07.2014" Zeit="22:00" Spielort="Salvador">
    <Mannschaft>Niederlande</Mannschaft>
    <Mannschaft>Costa Rica</Mannschaft>
    <Ergebnis ToreM1="4" ToreM2="3" Anmerkung="n.E."/>
  </Spiel>
</Runde>
...
<Runde Bezeichnung="Finale">
  <Spiel ID="F" Tag="13.07.2014" Zeit="21:00" Spielort="Rio d. J.">
    <Mannschaft>Deutschland</Mannschaft>
    <Mannschaft>Argentinien</Mannschaft>
    <Ergebnis ToreM1="1" ToreM2="0" Anmerkung="n.V."/>
  </Spiel>
</Runde>
</Spielplan>
```

(a) Ist das Dokument ein wohlgeformtes XML-Dokument? Wenn nein, welche Fehler enthält es?

(b) Bei Fußballspielen gibt es immer eine Reihenfolge der zwei Teilnehmer (Heim und Gast) und damit ist ein Ergebnis *x*:*y* immer eindeutig: *x* sind die von der ersten Mannschaft geschossenen Tore, *y* die der zweiten. Reicht die Aufzählung der beiden Mannschaft-Elemente im Dokument hierfür aus, oder muss man IDs einführen oder die Elemente umbenennen? Kurze Begründung.

Aufgabe 2:

Ergänzen Sie die unterstrichenen Lücken in der DTD für das (ggf. korrigierte) Spielplan-Dokument von oben! Es gilt die folgende Vorgabe: Die Attribute *Anmerkung* und *Spielort* sind optional, alle anderen Attribute sind Pflichtangaben. Das Attribut *ID* ist ein eindeutiger Bezeichner für das Spiel. Achten Sie auf leere Elemente! Bei *Anmerkung* steht „*n.V.*“ für „nach Verlängerung“, „*n.E.*“ für „nach Elfmeterschießen“, „*b.A.*“ für „bei Abbruch“. Einen Default-Wert gibt es nicht.

```
<!-- DTD fuer den Spielplan der Fussball-WM -->
<!ELEMENT Spielplan (Runde)*>
<!ATTLIST Spielplan Turnier _________________ #REQUIRED>

<!ELEMENT Runde (Spiel)+>
<!ATTLIST Runde Bezeichnung CDATA #REQUIRED>

<!ELEMENT Spiel (Mannschaft, Mannschaft, Ergebnis?)>
<!ATTLIST Spiel ________ _________ #REQUIRED
                Tag CDATA #REQUIRED
                Zeit CDATA #REQUIRED
                Spielort CDATA _________________>

<!ELEMENT Mannschaft (_____________________)>
<!ELEMENT Ergebnis _________________>
<!ATTLIST Ergebnis ToreM1 CDATA #REQUIRED
                   ToreM2 CDATA #REQUIRED
                   Anmerkung ( n.V. | n.E. | b.A. ) _________________>
```

Aufgabe 3:

Um Schreibarbeit zu sparen und Schreibfehler zu vermeiden, könnte man statt der Ländernamen in den Mannschaft-Elementen im Dokument nur Entitätsreferenzen eintragen, also z.B. `&D;` für Deutschland. Wie würde die Deklaration dieser Entität *D* in der DTD lauten?

```
<!_____________________________________________________________>
```

Aufgabe 4:

Ein XML-Schema zum Spielplan-Dokument sieht wie folgt aus. Füllen Sie die Lücken. Orientieren Sie sich bei den gewünschten Angaben an den Vorgaben der DTD aus Aufgabe 2. Zusätzlich bzw. abweichend soll gelten, dass die Attribute *ToreM1* und *ToreM2* im Element *Ergebnis* nicht-negative ganze Zahlen sind.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<xsd:element name="Spielplan">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Runde" type="____________________"
                   minOccurs="0" maxOccurs="____________________"/>
    </xsd:sequence>
    <xsd:attribute name="Turnier" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>

<xsd:complexType name="____________________">
  <xsd:sequence>
    <xsd:element name="Spiel" type="SpielType"
                 maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="Bezeichnung" type="xsd:string"
                 use="required"/>
</xsd:complexType>

<xsd:complexType name="SpielType">
  <xsd:sequence>
    <xsd:element name="Mannschaft" type="xsd:string"
```

```
                minOccurs="_____" maxOccurs="_____"/>
    <xsd:element name="Ergebnis" type="ErgebnisType"
                minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="_______" type="____________" use="required"/>
  <xsd:attribute name="Tag" type="xsd:string" use="required"/>
  <xsd:attribute name="Zeit" type="xsd:string" use="required"/>
  <xsd:attribute name="Spielort" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="ErgebnisType">
  <xsd:attribute name="ToreM1" type="_____________________________"
                use="required"/>
  <xsd:attribute name="ToreM2" type="_____________________________"
                use="required"/>
  <xsd:attribute name="Anmerkung">
    <xsd:simpleType>
      <xsd:_____________________ _____________________________>
        <xsd:enumeration value="n.V."/>
        <xsd:enumeration value="n.E."/>
        <xsd:enumeration value="b.A."/>
      </xsd:_____________________>
    </xsd:simpleType>
  </xsd:attribute>
</xsd:complexType>

</xsd:schema>
```

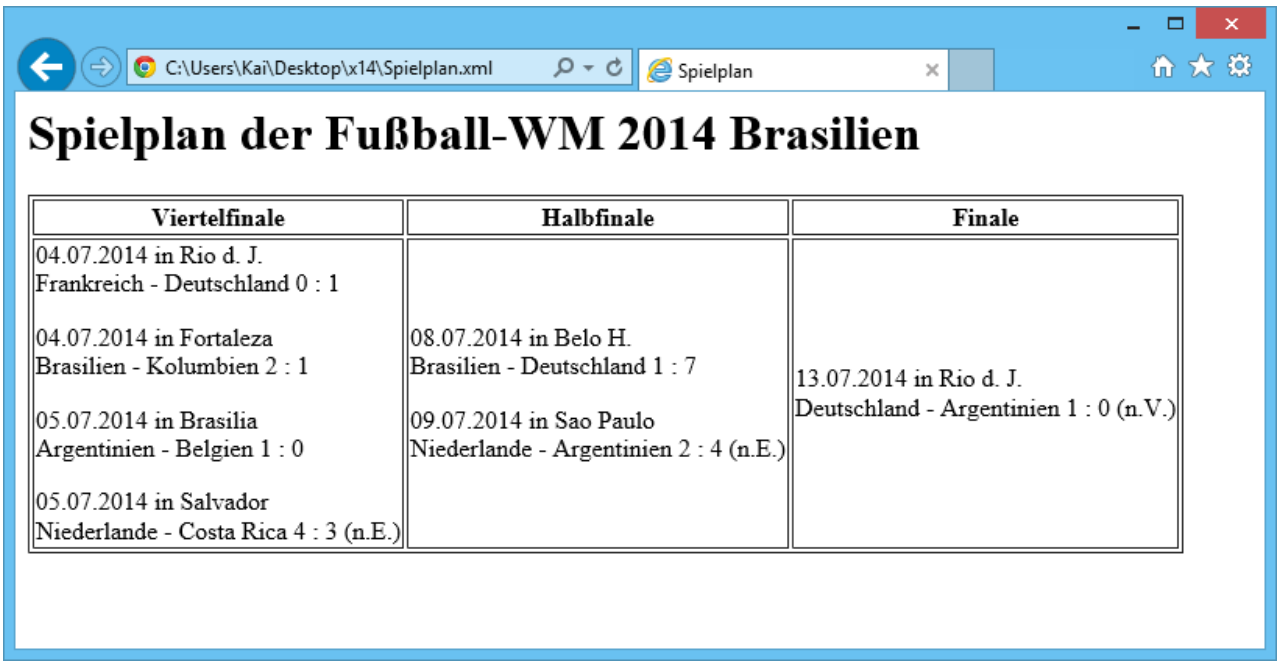
Aufgabe 5:

Ergänzen Sie die Lücken im Stylesheet, damit unser, ggf. korrigiertes Spielplan-Dokument (ohne Spiel um Platz 3) aus Aufgabe 1 die unten gezeigte Ausgabe erzeugt.

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet version='1.0'
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="____________________">
  <html>
    <head>
      <title>Spielplan</title>
    </head>
    <body>
      <h1>
        Spielplan der <xsl:value-of select="Spielplan/@Turnier"/>
      </h1>
      <table border="1">
        <tr>
          <xsl:for-each select="Spielplan/Runde">
            <th>
              <xsl:value-of select="_____________________________"/>
            </th>
          </xsl:for-each>
        </tr>
        <tr>
          <xsl:_____________________________ select="Spielplan/Runde"/>
        </tr>
      </table>
    </body>
  </html>
</xsl:template>

<xsl:template match="________________">
  <td>
    <xsl:apply-templates/>
  </td>
</xsl:template>

<xsl:template match="________________">
  <p>
```

```
    <xsl:value-of select="@Tag"/> in
    <xsl:value-of select="@Spielort"/>
    <br/>
    <xsl:value-of select="_____________________________"/> -
    <xsl:value-of select="_____________________________"/>
    <xsl:text> </xsl:text>
    <xsl:value-of select="_____________________________"/> :
    <xsl:value-of select="_____________________________"/>
    <xsl:if test="Ergebnis/@Anmerkung">
      (<xsl:value-of select="_____________________________"/>)
    </xsl:if>
  </p>
</xsl:template>

</xsl:stylesheet>
```

Ausgabe:



Aufgabe 6:

Beim Fußball ist keine Statistik zu blöd, als dass man sie nicht errechnen könnte. Wir suchen für jeden der beiden Finalteilnehmer die Anzahl der seit dem Viertelfinale geschossenen Tore einschließlich der im Finale geschossenen Tore (siehe Ausgabe Aufgabe 5). Füllen Sie die Lücken in der folgenden XQuery auf dem ggf. korrigierten Dokument **`Spielplan.xml`**, sodass das unten gezeigte Ergebnis geliefert wird?

```
<Torstatistik>
{
  let $doc := fn:doc("Spielplan.xml")
  for $m in $doc/Spielplan/Runde[@Bezeichnung="Finale"]/Spiel/Mannschaft
  return
    <Mannschaft Name="{ __________ }">
    {
      fn:sum($doc//Spiel[_____________________]/Ergebnis/___________) +
      fn:sum($doc//Spiel[_____________________]/Ergebnis/___________)
    }
    </Mannschaft>
}
</Torstatistik>
```

Ausgabe:

```
<Torstatistik>
  <Mannschaft Name="Deutschland">9</Mannschaft>
  <Mannschaft Name="Argentinien">5</Mannschaft>
</Torstatistik>
```

Aufgabe 7:

Gegeben sei die folgende Datenbanktabelle mit Namen **SPIELE**.

INRUNDE	MANNSCHAFT1	MANNSCHAFT2	AUSGANG
Viertelfinale	Argentinien	Belgien	1:0
Finale	Deutschland	Argentinien	1:0 n.V.
Viertelfinale	Brasilien	Kolumbien	2:1
Halbfinale	Brasilien	Deutschland	1:7
Viertelfinale	Niederlande	Costa Rica	4:3 n.E.
Viertelfinale	Frankreich	Deutschland	0:1
Halbfinale	Niederlande	Argentinien	2:4 n.E.
Platz3	Brasilien	Niederlande	0:3

Füllen Sie die Lücken in der SQL/XML-Abfrage auf der **SPIELE**-Tabelle, die als Ergebnis die unten gezeigte Ausgabe liefert. Hinweis: Die senkrechten Striche sind der SQL-Operator für die String-Verkettung.

```
SELECT ____________________(_____________________,

        ___________________________(_____________________________________),

        MANNSCHAFT1 || ' - ' || MANNSCHAFT2

       )

FROM SPIELE

WHERE INRUNDE = 'Viertelfinale'

ORDER BY MANNSCHAFT1;
```

Ausgabe:

```
<Spiel Ergebnis="1:0">Argentinien - Belgien</Spiel>
<Spiel Ergebnis="2:1">Brasilien - Kolumbien</Spiel>
<Spiel Ergebnis="0:1">Frankreich - Deutschland</Spiel>
<Spiel Ergebnis="4:3 n.E.">Niederlande - Costa Rica</Spiel>
```

Aufgabe 8:

In unserer Sammlung animierter Verkehrszeichen betrachten wir heute das Gefahrenzeichen 123 *Arbeitsstelle*.

(a) Ergänzen Sie die Lücken durch Angabe der passenden Ziffer! Hinweis: Nicht alle Ergänzungen werden gebraucht, manche aber mehrfach.

1 circle
2 indefinite
3 fill
4 oval
5 path
6 polygon
7 line

(b) Wenn sich der Oberkörper des Arbeiters dreht, ragt dann das Ende der Schaufel über den roten Rand des Schilds? Begründung!

```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg">

<rect x="0" y="0" width="600" height="600" fill="white"/>

<______ id="sandhill" style="stroke:none" fill="black"
  d="M 324 445 L 465 445 A 6 6 60 0 0 468 438 L 405 360
    A 20 20 0 0 0 375 360 L 355 390
    A 12 8 -45 0 1 343 400 L 322 438 A 6 6 60 0 0 324 445"/>

<line id="leftleg" x1="260" y1="330" x2="200" y2="440"
  style="stroke:black; stroke-width:20px; stroke-linecap:round"/>
<line id="rightupperleg" x1="270" y1="338" x2="290" y2="380"
  style="stroke:black; stroke-width:20px; stroke-linecap:round"/>
<line id="rightlowerleg" x1="290" y1="380" x2="290" y2="440"
  style="stroke:black; stroke-width:20px; stroke-linecap:round"/>

<g id="upperbody">
  <______ id="head" cx="330" cy="240" r="18" ______="black" stroke="none"/>
  <line id="rump" x1="267" y1="332" x2="300" y2="270"
```

```
    style="stroke:black; stroke-width:35px; stroke-linecap:round"/>
  <line id="leftarm" x1="308" y1="265" x2="308" y2="345"
    style="stroke:black; stroke-width:20px; stroke-linecap:round"/>
  <line id="rightarm1" x1="256" y1="263" x2="302" y2="263"
    style="stroke:black; stroke-width:20px; stroke-linecap:round"/>
  <line id="rightarm2" x1="256" y1="263" x2="234" y2="300"
    style="stroke:black; stroke-width:20px; stroke-linecap:round"/>
  <line id="shovel" x1="234" y1="295" x2="370" y2="385"
    style="stroke:black; stroke-width:8px; stroke-linecap:round"/>
  <______ id="shovel_end" style="stroke:none" fill="black"
    d="M 370 389 a 80 40 0 0 0 60 20 l -18 -30 a 12 8 0 0 0 -18 -4
      L 370 380 L 370 388"/>
  <line id="white-belt" x1="245" y1="320" x2="295" y2="348"
    style="stroke:white; stroke-width:8px"/>
  <animateTransform attributeName="transform" type="rotate"
    from="10 265 335" to="-33 265 335" begin="0s" dur="4s"
    repeatCount="____________"/>
</g>

<______ x1="300" y1="67" x2="550" y2="500"
  style="stroke:red; stroke-width:50px; stroke-linecap:round"/>
<______ x1="50" y1="500" x2="550" y2="500"
  style="stroke:red; stroke-width:50px; stroke-linecap:round"/>
<______ x1="50" y1="500" x2="300" y2="67"
  style="stroke:red; stroke-width:50px; stroke-linecap:round"/>

<______ style="stroke:black; stroke-width:2px" ______="none"
  d="M 260 50 A 43 43 60 0 1 340 50 L 590 480
    A 43 43 -60 0 1 550 545 L 50 545
    A 43 43 -60 0 1 10 480 L 260 50"/>

</svg>
```

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

MUSTERLÖSUNG

Nachname: Vorname:

Matr.Nr.: Studiengang:

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 90 Minuten.

Aufgabe	Punkte max.	Punkte erreicht
1	1 + 2	
2	4	
3	2	
4	6	
5	6	
6	4	
7	4	
8	4 + 1	
Summe	**34**	

Aufgabe 1:

Wir stehen immer noch unter dem Eindruck der Fußballweltmeisterschaft in Brasilien. Hier ist der Spielplan ab dem Viertelfinale (verkürzt).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="SpielplanStyle.xsl"?>
<!DOCTYPE Spielplan SYSTEM "Spielplan.dtd">
<Spielplan Turnier="Fußball-WM 2014 Brasilien">
<Runde Bezeichnung="Viertelfinale">
  <Spiel ID="VF1" Tag="04.07.2014" Zeit="18:00" Spielort="Rio d. J.">
    <Mannschaft>Frankreich</Mannschaft>
    <Mannschaft>Deutschland</Mannschaft>
    <Ergebnis ToreM1="0" ToreM2="1"/>
  </Spiel>
  <Spiel ID="VF2" Tag="04.07.2014" Zeit="22:00" Spielort="Fortaleza">
    <Mannschaft>Brasilien</Mannschaft>
    <Mannschaft>Kolumbien</Mannschaft>
    <Ergebnis ToreM1="2" ToreM2="1"/>
  </Spiel>
  <Spiel ID="VF3" Tag="05.07.2014" Zeit="18:00" Spielort="Brasilia">
    <Mannschaft>Argentinien</Mannschaft>
    <Mannschaft>Belgien</Mannschaft>
    <Ergebnis ToreM1="1" ToreM2="0"/>
  </Spiel>
  <Spiel ID="VF4" Tag="05.07.2014" Zeit="22:00" Spielort="Salvador">
    <Mannschaft>Niederlande</Mannschaft>
    <Mannschaft>Costa Rica</Mannschaft>
    <Ergebnis ToreM1="4" ToreM2="3" Anmerkung="n.E."/>
  </Spiel>
</Runde>
...
<Runde Bezeichnung="Finale">
  <Spiel ID="F" Tag="13.07.2014" Zeit="21:00" Spielort="Rio d. J.">
    <Mannschaft>Deutschland</Mannschaft>
    <Mannschaft>Argentinien</Mannschaft>
    <Ergebnis ToreM1="1" ToreM2="0" Anmerkung="n.V."/>
  </Spiel>
</Runde>
</Spielplan>
```

(a) Ist das Dokument ein wohlgeformtes XML-Dokument? Wenn nein, welche Fehler enthält es?

Das XML-Dokument ist wohlgeformt.

(b) Bei Fußballspielen gibt es immer eine Reihenfolge der zwei Teilnehmer (Heim und Gast) und damit ist ein Ergebnis *x*:*y* immer eindeutig: *x* sind die von der ersten Mannschaft geschossenen Tore, *y* die der zweiten. Reicht die Aufzählung der beiden Mannschaft-Elemente im Dokument hierfür aus, oder muss man IDs einführen oder die Elemente umbenennen? Kurze Begründung.

Elemente in XML-Dokumenten sind geordnet, damit ist die Reihenfolge der Mannschaft-Elemente eindeutig gegeben. Die Aufzählung reicht somit.

Aufgabe 2:

Ergänzen Sie die unterstrichenen Lücken in der DTD für das (ggf. korrigierte) Spielplan-Dokument von oben! Es gilt die folgende Vorgabe: Die Attribute *Anmerkung* und *Spielort* sind optional, alle anderen Attribute sind Pflichtangaben. Das Attribut *ID* ist ein eindeutiger Bezeichner für das Spiel. Achten Sie auf leere Elemente! Bei *Anmerkung* steht „*n.V.*“ für „nach Verlängerung“, „*n.E.*“ für „nach Elfmeterschießen“, „*b.A.*“ für „bei Abbruch“. Einen Default-Wert gibt es nicht.

```
<!-- DTD fuer den Spielplan der Fussball-WM -->
<!ELEMENT Spielplan (Runde)*>
<!ATTLIST Spielplan Turnier __CDATA__ #REQUIRED>

<!ELEMENT Runde (Spiel)+>
<!ATTLIST Runde Bezeichnung CDATA #REQUIRED>

<!ELEMENT Spiel (Mannschaft, Mannschaft, Ergebnis?)>
<!ATTLIST Spiel __ID__ __ID__ #REQUIRED
                Tag CDATA #REQUIRED
                Zeit CDATA #REQUIRED
                Spielort CDATA __#IMPLIED__>

<!ELEMENT Mannschaft (__#PCDATA__)>
<!ELEMENT Ergebnis __EMPTY__>
<!ATTLIST Ergebnis ToreM1 CDATA #REQUIRED
                   ToreM2 CDATA #REQUIRED
                   Anmerkung ( n.V. | n.E. | b.A. ) __#IMPLIED__>
```

Aufgabe 3:

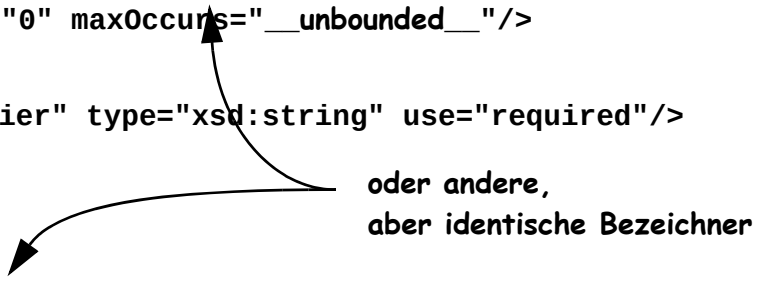
Um Schreibarbeit zu sparen und Schreibfehler zu vermeiden, könnte man statt der Ländernamen in den Mannschaft-Elementen im Dokument nur Entitätsreferenzen eintragen, also z.B. `&D;` für Deutschland. Wie würde die Deklaration dieser Entität *D* in der DTD lauten?

```
<!__ENTITY D "Deutschland"__>
```

Aufgabe 4:

Ein XML-Schema zum Spielplan-Dokument sieht wie folgt aus. Füllen Sie die Lücken. Orientieren Sie sich bei den gewünschten Angaben an den Vorgaben der DTD aus Aufgabe 2. Zusätzlich bzw. abweichend soll gelten, dass die Attribute *ToreM1* und *ToreM2* im Element *Ergebnis* nicht-negative ganze Zahlen sind.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<xsd:element name="Spielplan">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Runde" type="__RundeType__"
                   minOccurs="0" maxOccurs="__unbounded__"/>
    </xsd:sequence>
    <xsd:attribute name="Turnier" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>

<xsd:complexType name="__RundeType__">
  <xsd:sequence>
    <xsd:element name="Spiel" type="SpielType"
                 maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="Bezeichnung" type="xsd:string"
                 use="required"/>
</xsd:complexType>

<xsd:complexType name="SpielType">
  <xsd:sequence>
    <xsd:element name="Mannschaft" type="xsd:string"
```



```
                minOccurs="__2__" maxOccurs="__2__"/>
    <xsd:element name="Ergebnis" type="ErgebnisType"
                minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="__ID__" type="__xsd:ID__" use="required"/>
  <xsd:attribute name="Tag" type="xsd:string" use="required"/>
  <xsd:attribute name="Zeit" type="xsd:string" use="required"/>
  <xsd:attribute name="Spielort" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="ErgebnisType">
  <xsd:attribute name="ToreM1" type="__xsd:nonNegativeInteger__"
                use="required"/>
  <xsd:attribute name="ToreM2" type="__xsd:nonNegativeInteger__"
                use="required"/>
  <xsd:attribute name="Anmerkung">
    <xsd:simpleType>
      <xsd:__restriction__ __base="xsd:string"__>
        <xsd:enumeration value="n.V."/>
        <xsd:enumeration value="n.E."/>
        <xsd:enumeration value="b.A."/>
      </xsd:__restriction__>
    </xsd:simpleType>
  </xsd:attribute>
</xsd:complexType>

</xsd:schema>
```

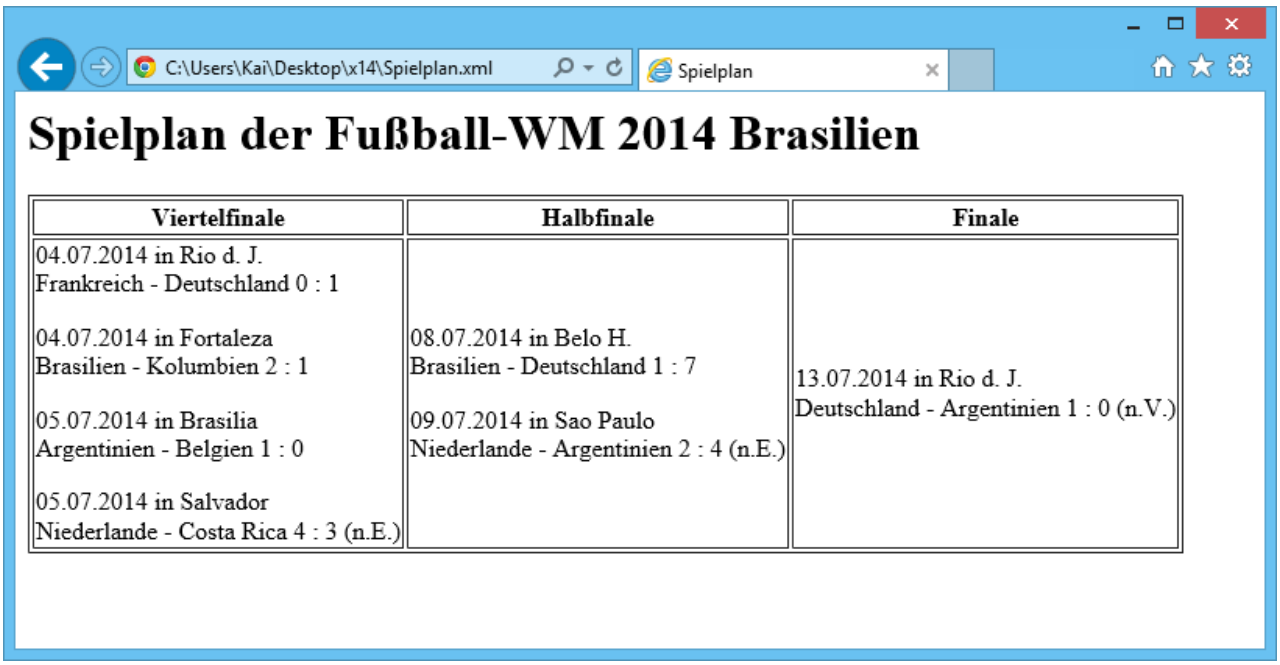
Aufgabe 5:

Ergänzen Sie die Lücken im Stylesheet, damit unser, ggf. korrigiertes Spielplan-Dokument (ohne Spiel um Platz 3) aus Aufgabe 1 die unten gezeigte Ausgabe erzeugt.

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet version='1.0'
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="__/__">
  <html>
    <head>
      <title>Spielplan</title>
    </head>
    <body>
      <h1>
        Spielplan der <xsl:value-of select="Spielplan/@Turnier"/>
      </h1>
      <table border="1">
        <tr>
          <xsl:for-each select="Spielplan/Runde">
            <th>
              <xsl:value-of select="__@Bezeichnung__"/>
            </th>
          </xsl:for-each>
        </tr>
        <tr>
          <xsl:__apply-templates__ select="Spielplan/Runde"/>
        </tr>
      </table>
    </body>
  </html>
</xsl:template>

<xsl:template match="__Runde__">
  <td>
    <xsl:apply-templates/>
  </td>
</xsl:template>

<xsl:template match="__Spiel__">
  <p>
```

```
    <xsl:value-of select="@Tag"/> in
    <xsl:value-of select="@Spielort"/>
    <br/>
    <xsl:value-of select="__Mannschaft[1]__"/> -
    <xsl:value-of select="__Mannschaft[2]__"/>
    <xsl:text> </xsl:text>
    <xsl:value-of select="__Ergebnis/@ToreM1__"/> :
    <xsl:value-of select="__Ergebnis/@ToreM2__"/>
    <xsl:if test="Ergebnis/@Anmerkung">
      (<xsl:value-of select="__Ergebnis/@Anmerkung__"/>)
    </xsl:if>
  </p>
</xsl:template>

</xsl:stylesheet>
```

Ausgabe:



Aufgabe 6:

Beim Fußball ist keine Statistik zu blöd, als dass man sie nicht errechnen könnte. Wir suchen für jeden der beiden Finalteilnehmer die Anzahl der seit dem Viertelfinale geschossenen Tore einschließlich der im Finale geschossenen Tore (siehe Ausgabe Aufgabe 5). Füllen Sie die Lücken in der folgenden XQuery auf dem ggf. korrigierten Dokument **`Spielplan.xml`**, sodass das unten gezeigte Ergebnis geliefert wird?

```
<Torstatistik>
{
  let $doc := fn:doc("Spielplan.xml")
  for $m in $doc/Spielplan/Runde[@Bezeichnung="Finale"]/Spiel/Mannschaft
  return
    <Mannschaft Name="{ __$m__ }">
    {
      fn:sum($doc//Spiel[__Mannschaft[1]=$m__]/Ergebnis/__@ToreM1__) +
      fn:sum($doc//Spiel[__Mannschaft[2]=$m__]/Ergebnis/__@ToreM2__)
    }
    </Mannschaft>
}
</Torstatistik>
```

Ausgabe:

```
<Torstatistik>
  <Mannschaft Name="Deutschland">9</Mannschaft>
  <Mannschaft Name="Argentinien">5</Mannschaft>
</Torstatistik>
```

Aufgabe 7:

Gegeben sei die folgende Datenbanktabelle mit Namen **SPIELE**.

INRUNDE	MANNSCHAFT1	MANNSCHAFT2	AUSGANG
Viertelfinale	Argentinien	Belgien	1:0
Finale	Deutschland	Argentinien	1:0 n.V.
Viertelfinale	Brasilien	Kolumbien	2:1
Halbfinale	Brasilien	Deutschland	1:7
Viertelfinale	Niederlande	Costa Rica	4:3 n.E.
Viertelfinale	Frankreich	Deutschland	0:1
Halbfinale	Niederlande	Argentinien	2:4 n.E.
Platz3	Brasilien	Niederlande	0:3

Füllen Sie die Lücken in der SQL/XML-Abfrage auf der **SPIELE**-Tabelle, die als Ergebnis die unten gezeigte Ausgabe liefert. Hinweis: Die senkrechten Striche sind der SQL-Operator für die String-Verkettung.

```
SELECT __XMLELEMENT__(__NAME "Spiel"__,
         __XMLATTRIBUTES__(__AUSGANG AS "Ergebnis"__),
         MANNSCHAFT1 || ' - ' || MANNSCHAFT2
       )
FROM SPIELE
WHERE INRUNDE = 'Viertelfinale'
ORDER BY MANNSCHAFT1;
```

Ausgabe:

```
<Spiel Ergebnis="1:0">Argentinien - Belgien</Spiel>
<Spiel Ergebnis="2:1">Brasilien - Kolumbien</Spiel>
<Spiel Ergebnis="0:1">Frankreich - Deutschland</Spiel>
<Spiel Ergebnis="4:3 n.E.">Niederlande - Costa Rica</Spiel>
```

Aufgabe 8:

In unserer Sammlung animierter Verkehrszeichen betrachten wir heute das Gefahrenzeichen 123 *Arbeitsstelle*.

(a) Ergänzen Sie die Lücken durch Angabe der passenden Ziffer! Hinweis: Nicht alle Ergänzungen werden gebraucht, manche aber mehrfach.

1 circle
2 indefinite
3 fill
4 oval
5 path
6 polygon
7 line

(b) Wenn sich der Oberkörper des Arbeiters dreht, ragt dann das Ende der Schaufel über den roten Rand des Schilds? Begründung!

__Nein, da roter Rand zuletzt gezeichnet wird.__

```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg">

<rect x="0" y="0" width="600" height="600" fill="white"/>

<__5__ id="sandhill" style="stroke:none" fill="black"
  d="M 324 445 L 465 445 A 6 6 60 0 0 468 438 L 405 360
    A 20 20 0 0 0 375 360 L 355 390
    A 12 8 -45 0 1 343 400 L 322 438 A 6 6 60 0 0 324 445"/>

<line id="leftleg" x1="260" y1="330" x2="200" y2="440"
  style="stroke:black; stroke-width:20px; stroke-linecap:round"/>
<line id="rightupperleg" x1="270" y1="338" x2="290" y2="380"
  style="stroke:black; stroke-width:20px; stroke-linecap:round"/>
<line id="rightlowerleg" x1="290" y1="380" x2="290" y2="440"
  style="stroke:black; stroke-width:20px; stroke-linecap:round"/>

<g id="upperbody">
  <__1__ id="head" cx="330" cy="240" r="18" __3__="black" stroke="none"/>
  <line id="rump" x1="267" y1="332" x2="300" y2="270"
```

```
    style="stroke:black; stroke-width:35px; stroke-linecap:round"/>
  <line id="leftarm" x1="308" y1="265" x2="308" y2="345"
    style="stroke:black; stroke-width:20px; stroke-linecap:round"/>
  <line id="rightarm1" x1="256" y1="263" x2="302" y2="263"
    style="stroke:black; stroke-width:20px; stroke-linecap:round"/>
  <line id="rightarm2" x1="256" y1="263" x2="234" y2="300"
    style="stroke:black; stroke-width:20px; stroke-linecap:round"/>
  <line id="shovel" x1="234" y1="295" x2="370" y2="385"
    style="stroke:black; stroke-width:8px; stroke-linecap:round"/>
  <__5__ id="shovel_end" style="stroke:none" fill="black"
    d="M 370 389 a 80 40 0 0 0 60 20 l -18 -30 a 12 8 0 0 0 -18 -4
      L 370 380 L 370 388"/>
  <line id="white-belt" x1="245" y1="320" x2="295" y2="348"
    style="stroke:white; stroke-width:8px"/>
  <animateTransform attributeName="transform" type="rotate"
    from="10 265 335" to="-33 265 335" begin="0s" dur="4s"
    repeatCount="__2__"/>
</g>

<__7__ x1="300" y1="67" x2="550" y2="500"
  style="stroke:red; stroke-width:50px; stroke-linecap:round"/>
<__7__ x1="50" y1="500" x2="550" y2="500"
  style="stroke:red; stroke-width:50px; stroke-linecap:round"/>
<__7__ x1="50" y1="500" x2="300" y2="67"
  style="stroke:red; stroke-width:50px; stroke-linecap:round"/>

<__5__ style="stroke:black; stroke-width:2px" __3__="none"
  d="M 260 50 A 43 43 60 0 1 340 50 L 590 480
    A 43 43 -60 0 1 550 545 L 50 545
    A 43 43 -60 0 1 10 480 L 260 50"/>

</svg>
```

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

Nachname: Vorname:

Matr.Nr.: Studiengang:

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 90 Minuten.

Aufgabe	Punkte max.	Punkte erreicht
1	1 + 2	
2	4	
3	2	
4	6	
5	6	
6	4	
7	4	
8	4 + 2	
Summe	**35**	

Aufgabe 1:

Für diese XML-Klausur haben wir uns etwas Einfaches ausgedacht: eine fiktive Aufstellung von Handytarifen.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="HandytarifeStyle.xsl"?>
<!DOCTYPE Tarifanbieter SYSTEM "Handytarife.dtd">
<Tarifanbieter Stand="2015-02-24">
  <Angebot Vertragsart="prepaid">
    <Anbieter Tarif="Prepaid +100" Netz="D1">
      Kingster
    </Anbieter>
    <Preise einmalig="9.99" Anruf="9.0" SMS="9.0" Internet="24.0"/>
  </Angebot>

  <Angebot Vertragsart="Laufzeit">
    <Anbieter Tarif="Smart plus" Netz="o2">
      Select Me
    </Anbieter>
    <Preise monatlich="9.95" Anruf="7.0" SMS="8.0" Internet="19.0"/>
    <Vertragsdauer>24 Monate</Vertragsdauer>
  </Angebot>

  <Angebot Vertragsart="prepaid">
    <Anbieter Tarif="9Cent prepaid" Netz="E-Plus">
      green.de
    </Anbieter>
    <Preise einmalig="9.90" Anruf="9.0" SMS="9.0" Internet="24.0"/>
    <Aufladung>Online autom. Bargeld Anruf&amp;amp;SMS</Aufladung>
  </Angebot>

  <Angebot Vertragsart="prepaid">
    <Anbieter Tarif="MrSpar" Netz="D2">
      MisterHandy
    </Anbieter>
    <Preise einmalig="9.95" Anruf="8.0" SMS="8.0" Internet="49.0"/>
    <Aufladung>Bankeinzug autom. Anruf&amp;amp;SMS</Aufladung>
  </Angebot>
</Tarifanbieter>
```

(a) Ist das Dokument ein wohlgeformtes XML-Dokument? Wenn nein, welche Fehler enthält es und korrigieren Sie diese.

(b) Im Dokument wird die vordefinierte Entität **`&`** verwendet. Nennen Sie zwei weitere vordefinierte Entitäten!

Aufgabe 2:

Ergänzen Sie die unterstrichenen Lücken in der DTD für das (ggf. korrigierte) Handytarife-Dokument von oben! Es gilt die folgende Vorgabe:

- **`Tarifanbieter`** enthält beliebig viele, mindestens aber ein **`Angebot`**.
- Das Attribut **`Vertragsart`** ist ein Aufzählungstyp mit den Werten **`prepaid`** und **`Laufzeit`**, wobei **`prepaid`** der Standardwert ist.
- **`Preise`** darf keine Unterelemente und keinen Text enthalten.
- Die Attribute **`einmalig`** und **`monatlich`** sind optional.

```
<!-- DTD fuer die Handytarife -->
<!ELEMENT Tarifanbieter _____________________________>
<!ATTLIST Tarifanbieter Stand CDATA #REQUIRED>

<!ELEMENT Angebot (Anbieter, Preise, (Aufladung | Vertragsdauer)?)>
<!ATTLIST Angebot Vertragsart _____________________________________>

<!ELEMENT Anbieter (#PCDATA)>
<!ATTLIST Anbieter Tarif CDATA #REQUIRED
                   Netz CDATA #REQUIRED>

<!ELEMENT Preise ____________________>
<!ATTLIST Preise einmalig CDATA _____________________
                 monatlich CDATA _____________________
                 Anruf CDATA #REQUIRED
                 SMS CDATA #REQUIRED
                 Internet CDATA #REQUIRED>
<!ELEMENT Aufladung (#PCDATA)>
<!ELEMENT Vertragsdauer (#PCDATA)>
```

Aufgabe 3:

Kann man in der DTD oben sicherstellen, dass bei gegebenem Attributwert **`prepaid`** für **`Vertragsart`** kein Element **`Vertragsdauer`** im Dokument erscheint? Wenn ja, wie?

Aufgabe 4:

Ein XML-Schema zum Handytarife-Dokument sieht wie folgt aus. Füllen Sie die Lücken. Richten Sie sich bei den gewünschten Angaben an den Vorgaben der DTD aus Aufgabe 2. Zusätzlich bzw. abweichend soll gelten, dass das Attribut **`Stand`** ein Datumstyp ist.

```
<?xml version="1.0"?>

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


  <xsd:element name="Tarifanbieter" type="TarifanbieterType"/>


  <xsd:complexType name="TarifanbieterType">
    <xsd:sequence>
      <xsd:element name="Angebot" type="AngebotType"
            maxOccurs="_____________________"/>
    </xsd:sequence>
    <xsd:attribute name="Stand" type="____________" use="required"/>
  </xsd:complexType>


  <xsd:complexType name="AngebotType">
    <xsd:sequence>
      <xsd:element name="Anbieter" type="AnbieterType"/>
      <xsd:element name="Preise" type="_____________________"/>
      <xsd:_______________ minOccurs="0">
        <xsd:element name="Aufladung" type="xsd:string"/>
        <xsd:element name="Vertragsdauer" type="xsd:string"/>
      </xsd:_______________>
    </xsd:sequence>
```

```
    <xsd:attribute name="Vertragsart" default="prepaid">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:_____________________ value="prepaid"/>
          <xsd:_____________________ value="Laufzeit"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>

  <xsd:complexType name="AnbieterType">
    <xsd:_____________________________>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="Tarif" type="xsd:string"
            use="required"/>
        <xsd:attribute name="Netz" type="xsd:string" use="required"/>
      </xsd:extension>
    </xsd:_____________________________>
  </xsd:complexType>

  <xsd:complexType name="_____________________">
    <xsd:attribute name="einmalig" type="xsd:decimal"/>
    <xsd:attribute name="monatlich" type="xsd:decimal"/>
    <xsd:attribute name="Anruf" type="xsd:decimal" use="required"/>
    <xsd:attribute name="SMS" type="xsd:decimal" use="required"/>
    <xsd:attribute name="Internet" type="xsd:decimal" use="required"/>
  </xsd:complexType>

</xsd:schema>
```

Aufgabe 5:

Ergänzen Sie die Lücken im Stylesheet, damit unser, ggf. korrigiertes Handytarife-Dokument aus Aufgabe 1 die unten gezeigte tabellarische Ausgabe der **prepaid**-Tarife erzeugt.

```
<?xml version='1.0'?>
<xsl:stylesheet version='1.0'
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:variable name="_____________________">prepaid</xsl:variable>

<xsl:template match="/">
  <html>
  <head><title>Handytarife</title></head>
  <body>
    <h1>
      Handytarife Vertragsart
      <xsl:_____________________ select="$Auswahl"/>
    </h1>
    <h2>
      Anzahl der Angebote:
      <xsl:value-of
        select="count(//Angebot[_____________________________________])"/>
    </h2>
    <table border="1" cellspacing="0" cellpadding="10">
      <tr>
        <th>Anbieter und Tarif</th><th>Netz</th><th>Preise</th>
      </tr>
      <xsl:apply-templates
        select="//Angebot[_____________________________________]"/>
    </table>
  </body>
  </html>
</xsl:template>

<xsl:template _____________________________________>
  <tr>
    <td>
      <xsl:value-of select="Anbieter"/><br/>
      <xsl:value-of select="_____________________________________"/>
    </td>
    <td><img src="{Anbieter/@Netz}.jpg"/></td>
    <td>
```

```
      <xsl:choose>
        <xsl:when _________________="@Vertragsart='prepaid'">
          einmalig: <xsl:value-of
            select="_____________________________"/> Euro<br/>
        </xsl:when>
        <xsl:when _________________="@Vertragsart='Laufzeit'">
          monatlich: <xsl:value-of
            select="_____________________________"/> Euro<br/>
        </xsl:when>
      </xsl:choose>
      Anruf: <xsl:value-of select="Preise/@Anruf"/> Cent<br/>
      SMS: <xsl:value-of select="Preise/@SMS"/> Cent<br/>
      Internet je MB: <xsl:value-of select="Preise/@Internet"/> Cent
    </td>
  </tr>
</xsl:template>

</xsl:stylesheet>
```

Ausgabe:

Aufgabe 6:

Für unsere fiktive Liste der Handytarife berechnen wir Beispielkosten je Angebot auf der Basis der in der XQuery angegebenen Nutzung. Beachten Sie, dass die Preise im Dokument hierfür in Cent sind (vgl. Abbildung in Aufgabe 5) und wir einmalige und monatliche Fixkosten nicht einrechnen. Wie lautet die Ausgabe für das ggf. korrigierte Dokument **`Handytarife.xml`** aus Aufgabe 1?

```
<Beispielkosten>
{
  let $doc := fn:doc("Handytarife.xml")
  for $v in fn:distinct-values($doc/Tarifanbieter/Angebot/@Vertragsart)
  return
    <Vertragsart Typ="{$v}">
    {
      for $a in $doc/Tarifanbieter/Angebot[@Vertragsart=$v]
      return
        <Tarif Name="{$a/Anbieter/@Tarif}">
        {
          (10 * $a/Preise/@Anruf + 100 * $a/Preise/@Internet) div 100
        }
        </Tarif>
    }
    </Vertragsart>
}
</Beispielkosten>
```

Ausgabe:

```
<Beispielkosten>
```

```
</Beispielkosten>
```

Aufgabe 7:

Gegeben sei die folgende Datenbanktabelle mit Namen **TARIFE**.

ANBIETER	TARIF	NETZ	ANMERKUNG
Vodafone	CallYa Allnet	D2	prepaid
O2	Loop Smart L	o2	prepaid
Fonic	Fonic Smart	o2	500 Freieinheiten
freenetmobile	freeflat	D1	24 Monate Laufzeit

Füllen Sie die Lücken in der SQL/XML-Abfrage auf der **TARIFE**-Tabelle, die als Ergebnis die unten gezeigte Ausgabe liefert.

```
SELECT XMLELEMENT(
         NAME "Netz",
         XMLATTRIBUTES(_____________________________),
         XMLAGG(
           XMLELEMENT(
             _____________________,
             XMLATTRIBUTES(ANBIETER AS "Anbieter"),
             _________________
           )
         )
       )
FROM TARIFE
GROUP BY _________________
ORDER BY NETZ;
```

Ausgabe:

```
<Netz Netzbetreiber="D1">
  <Tarif Anbieter="freenetmobile">freeflat</Tarif>
</Netz>
<Netz Netzbetreiber="D2">
  <Tarif Anbieter="Vodafone">CallYa Allnet</Tarif>
</Netz>
<Netz Netzbetreiber="o2">
  <Tarif Anbieter="O2">Loop Smart L</Tarif>
  <Tarif Anbieter="Fonic">Fonic Smart</Tarif>
</Netz>
```

Aufgabe 8:

In unserer Sammlung animierter Verkehrszeichen betrachten wir heute das Gefahrenzeichen 128 *Bewegliche Brücke*.

(a) Ergänzen Sie die Lücken durch Angabe der passenden Ziffer! Hinweis: Nicht alle Ergänzungen werden gebraucht, manche aber mehrfach.

1 rect
2 repeatCount
3 fill
4 svg
5 path
6 points
7 line

```
<?xml version="1.0"?>
<_________ xmlns="http://www.w3.org/2000/svg">
<rect x="0" y="0" width="600" height="600"
  fill="white"/>

<_________ x1="50" y1="500" x2="550" y2="500"
  style="stroke:red; stroke-width:50px;
  stroke-linecap:round"/>
<_________ x1="50" y1="500" x2="300" y2="67"
  style="stroke:red; stroke-width:50px;
  stroke-linecap:round"/>
<_________ x1="300" y1="67" x2="550" y2="500"
  style="stroke:red; stroke-width:50px;
  stroke-linecap:round"/>
<path style="stroke:black; stroke-width:2px" fill="none"
  d="M 260 50 A 43 43 60 0 1 340 50 L 590 480
    A 43 43 -60 0 1 550 545 L 50 545
    A 43 43 -60 0 1 10 480 L 260 50"/>

<polygon _________="110,465 168,368 220,368 220,465"
  style="fill:black; stroke:none"/>
```

```
<polygon _________="380,465 380,368 432,368 486,465"
  style="fill:black; stroke:none"/>

<_________ x="220" y="348" width="160" height="20" fill="black">
  <animateTransform attributeName="transform" type="rotate"
    values="0 220 368; -40 220 368; 0 220 368" dur="5s"
    _________="indefinite"/>
</_________>

<path style="stroke:black; stroke-width:15px" fill="none"
  d="M 225,420 q 16,-10 32,0 t 32,0"/>
<path style="stroke:black; stroke-width:15px" fill="none"
  d="M 288,421 q 16,-10 32,0 t 32,0"/>
<path style="stroke:black; stroke-width:15px" fill="none"
  d="M 225,450 q 16,-10 32,0 t 32,0"/>
<path style="stroke:black; stroke-width:15px" fill="none"
  d="M 288,451 q 16,-10 32,0 t 32,0"/>
<path style="stroke:black; stroke-width:15px" fill="none"
  d="M 351,422 q 16,-10 32,0 t 32,0"/>
<path style="stroke:black; stroke-width:15px" fill="none"
  d="M 351,452 q 16,-10 32,0 t 32,0"/>
<_________ x1="225" y1="380" x2="225" y2="470"
  style="stroke:white; stroke-width:10px"/>
<_________ x1="375" y1="380" x2="375" y2="470"
  style="stroke:white; stroke-width:10px"/>
</_________>
```

(b) Wie wäre das **`values`**-Attribut im **`animateTransform`**-Element zu ändern, damit der Ankerpunkt (Drehpunkt) der Brücke rechts unten liegt, die Brücke aber gleich hoch öffnet?

`values="`_____**`"`**

ENDE DER KLAUSUR

Klausur zur Vorlesung „Einführung in XML“

MUSTERLÖSUNG

Nachname: Vorname:

Matr.Nr.: Studiengang:

Bearbeiten Sie alle Aufgaben! Hilfsmittel sind nicht zugelassen. Die Bearbeitungszeit ist 90 Minuten.

Aufgabe	Punkte max.	Punkte erreicht
1	1 + 2	
2	4	
3	2	
4	6	
5	6	
6	4	
7	4	
8	4 + 2	
Summe	**35**	

Aufgabe 1:

Für diese XML-Klausur haben wir uns etwas Einfaches ausgedacht: eine fiktive Aufstellung von Handytarifen.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="HandytarifeStyle.xsl"?>
<!DOCTYPE Tarifanbieter SYSTEM "Handytarife.dtd">
<Tarifanbieter Stand="2015-02-24">
  <Angebot Vertragsart="prepaid">
    <Anbieter Tarif="Prepaid +100" Netz="D1">
      Kingster
    </Anbieter>
    <Preise einmalig="9.99" Anruf="9.0" SMS="9.0" Internet="24.0"/>
  </Angebot>

  <Angebot Vertragsart="Laufzeit">
    <Anbieter Tarif="Smart plus" Netz="o2">
      Select Me
    </Anbieter>
    <Preise monatlich="9.95" Anruf="7.0" SMS="8.0" Internet="19.0"/>
    <Vertragsdauer>24 Monate</Vertragsdauer>
  </Angebot>

  <Angebot Vertragsart="prepaid">
    <Anbieter Tarif="9Cent prepaid" Netz="E-Plus">
      green.de
    </Anbieter>
    <Preise einmalig="9.90" Anruf="9.0" SMS="9.0" Internet="24.0"/>
    <Aufladung>Online autom. Bargeld Anruf&amp;amp;SMS</Aufladung>
  </Angebot>

  <Angebot Vertragsart="prepaid">
    <Anbieter Tarif="MrSpar" Netz="D2">
      MisterHandy
    </Anbieter>
    <Preise einmalig="9.95" Anruf="8.0" SMS="8.0" Internet="49.0"/>
    <Aufladung>Bankeinzug autom. Anruf&amp;amp;SMS</Aufladung>
  </Angebot>
</Tarifanbieter>
```

(a) Ist das Dokument ein wohlgeformtes XML-Dokument? Wenn nein, welche Fehler enthält es und korrigieren Sie diese.

Ja, das Dokument ist wohlgeformt.

(b) Im Dokument wird die vordefinierte Entität **&** verwendet. Nennen Sie zwei weitere vordefinierte Entitäten!

" ' < >

Aufgabe 2:

Ergänzen Sie die unterstrichenen Lücken in der DTD für das (ggf. korrigierte) Handytarife-Dokument von oben! Es gilt die folgende Vorgabe:

- **Tarifanbieter** enthält beliebig viele, mindestens aber ein **Angebot**.
- Das Attribut **Vertragsart** ist ein Aufzählungstyp mit den Werten **prepaid** und **Laufzeit**, wobei **prepaid** der Standardwert ist.
- **Preise** darf keine Unterelemente und keinen Text enthalten.
- Die Attribute **einmalig** und **monatlich** sind optional.

```
<!-- DTD fuer die Handytarife -->
<!ELEMENT Tarifanbieter ____(Angebot+)_____>
<!ATTLIST Tarifanbieter Stand CDATA #REQUIRED>

<!ELEMENT Angebot (Anbieter, Preise, (Aufladung | Vertragsdauer)?)>
<!ATTLIST Angebot Vertragsart __(prepaid | Laufzeit) "prepaid"__>

<!ELEMENT Anbieter (#PCDATA)>
<!ATTLIST Anbieter Tarif CDATA #REQUIRED
                   Netz CDATA #REQUIRED>

<!ELEMENT Preise ___EMPTY___>
<!ATTLIST Preise einmalig CDATA __#IMPLIED___
                 monatlich CDATA ___#IMPLIED___
                 Anruf CDATA #REQUIRED
                 SMS CDATA #REQUIRED
                 Internet CDATA #REQUIRED>
<!ELEMENT Aufladung (#PCDATA)>
<!ELEMENT Vertragsdauer (#PCDATA)>
```

Aufgabe 3:

Kann man in der DTD oben sicherstellen, dass bei gegebenem Attributwert **prepaid** für **Vertragsart** kein Element **Vertragsdauer** im Dokument erscheint? Wenn ja, wie?

Nein, das ist nicht möglich.

Aufgabe 4:

Ein XML-Schema zum Handytarife-Dokument sieht wie folgt aus. Füllen Sie die Lücken. Richten Sie sich bei den gewünschten Angaben an den Vorgaben der DTD aus Aufgabe 2. Zusätzlich bzw. abweichend soll gelten, dass das Attribut **Stand** ein Datumstyp ist.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:element name="Tarifanbieter" type="TarifanbieterType"/>

  <xsd:complexType name="TarifanbieterType">
    <xsd:sequence>
      <xsd:element name="Angebot" type="AngebotType"
            maxOccurs="__unbounded__"/>
    </xsd:sequence>
    <xsd:attribute name="Stand" type="__xsd:date__" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="AngebotType">
    <xsd:sequence>
      <xsd:element name="Anbieter" type="AnbieterType"/>
      <xsd:element name="Preise" type="__PreiseType__"/>
      <xsd:__choice__ minOccurs="0">
        <xsd:element name="Aufladung" type="xsd:string"/>
        <xsd:element name="Vertragsdauer" type="xsd:string"/>
      </xsd:__choice__>
    </xsd:sequence>
```

```
    <xsd:attribute name="Vertragsart" default="prepaid">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:__enumeration__ value="prepaid"/>
          <xsd:__enumeration__ value="Laufzeit"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>

  <xsd:complexType name="AnbieterType">
    <xsd:__simpleContent__>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="Tarif" type="xsd:string"
            use="required"/>
        <xsd:attribute name="Netz" type="xsd:string" use="required"/>
      </xsd:extension>
    </xsd:__simpleContent__>
  </xsd:complexType>

  <xsd:complexType name="___PreiseType___">
    <xsd:attribute name="einmalig" type="xsd:decimal"/>
    <xsd:attribute name="monatlich" type="xsd:decimal"/>
    <xsd:attribute name="Anruf" type="xsd:decimal" use="required"/>
    <xsd:attribute name="SMS" type="xsd:decimal" use="required"/>
    <xsd:attribute name="Internet" type="xsd:decimal" use="required"/>
  </xsd:complexType>

</xsd:schema>
```

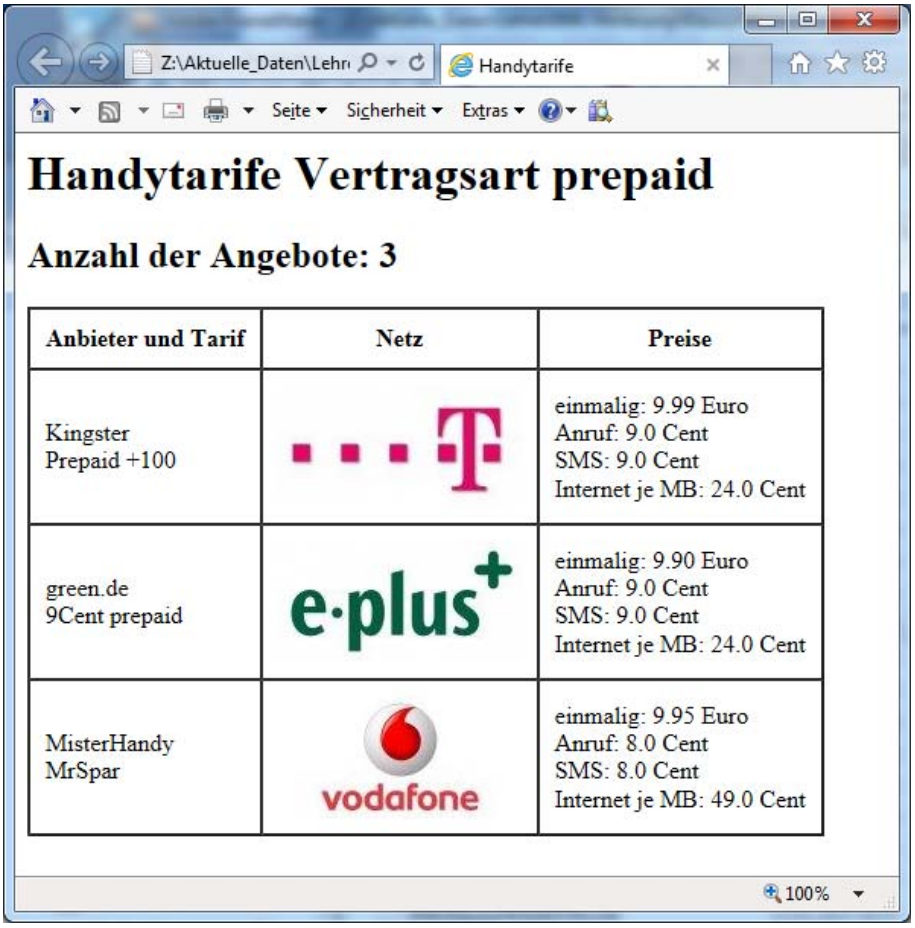
Aufgabe 5:

Ergänzen Sie die Lücken im Stylesheet, damit unser, ggf. korrigiertes Handytarife-Dokument aus Aufgabe 1 die unten gezeigte tabellarische Ausgabe der **prepaid**-Tarife erzeugt.

```
<?xml version='1.0'?>
<xsl:stylesheet version='1.0'
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:variable name="__Auswahl__">prepaid</xsl:variable>

<xsl:template match="/">
  <html>
  <head><title>Handytarife</title></head>
  <body>
    <h1>
      Handytarife Vertragsart
      <xsl:__value-of__ select="$Auswahl"/>
    </h1>
    <h2>
      Anzahl der Angebote:
      <xsl:value-of
        select="count(//Angebot[__@Vertragsart=$Auswahl__])"/>
    </h2>
    <table border="1" cellspacing="0" cellpadding="10">
      <tr>
        <th>Anbieter und Tarif</th><th>Netz</th><th>Preise</th>
      </tr>
      <xsl:apply-templates
        select="//Angebot[__@Vertragsart=$Auswahl__]"/>
    </table>
  </body>
  </html>
</xsl:template>

<xsl:template ___match="Angebot"___>
  <tr>
    <td>
      <xsl:value-of select="Anbieter"/><br/>
      <xsl:value-of select="___Anbieter/@Tarif___"/>
    </td>
    <td><img src="{Anbieter/@Netz}.jpg"/></td>
    <td>
```

```
      <xsl:choose>
        <xsl:when __test__="@Vertragsart='prepaid'">
          einmalig: <xsl:value-of
            select="__Preise/@einmalig__"/> Euro<br/>
        </xsl:when>
        <xsl:when __test__="@Vertragsart='Laufzeit'">
          monatlich: <xsl:value-of
            select="__Preise/@monatlich__"/> Euro<br/>
        </xsl:when>
      </xsl:choose>
      Anruf: <xsl:value-of select="Preise/@Anruf"/> Cent<br/>
      SMS: <xsl:value-of select="Preise/@SMS"/> Cent<br/>
      Internet je MB: <xsl:value-of select="Preise/@Internet"/> Cent
    </td>
  </tr>
</xsl:template>

</xsl:stylesheet>
```

Ausgabe:



Aufgabe 6:

Für unsere fiktive Liste der Handytarife berechnen wir Beispielkosten je Angebot auf der Basis der in der XQuery angegebenen Nutzung. Beachten Sie, dass die Preise im Dokument hierfür in Cent sind (vgl. Abbildung in Aufgabe 5) und wir einmalige und monatliche Fixkosten nicht einrechnen. Wie lautet die Ausgabe für das ggf. korrigierte Dokument **`Handytarife.xml`** aus Aufgabe 1?

```
<Beispielkosten>
{
  let $doc := fn:doc("Handytarife.xml")
  for $v in fn:distinct-values($doc/Tarifanbieter/Angebot/@Vertragsart)
  return
    <Vertragsart Typ="{$v}">
    {
      for $a in $doc/Tarifanbieter/Angebot[@Vertragsart=$v]
      return
        <Tarif Name="{$a/Anbieter/@Tarif}">
        {
          (10 * $a/Preise/@Anruf + 100 * $a/Preise/@Internet) div 100
        }
        </Tarif>
    }
    </Vertragsart>
}
</Beispielkosten>
```

Ausgabe:

```
<Beispielkosten>
  <Vertragsart Typ="prepaid">
    <Tarif Name="Prepaid +100">24.9</Tarif>
    <Tarif Name="9Cent prepaid">24.9</Tarif>
    <Tarif Name="MrSpar">49.8</Tarif>
  </Vertragsart>
  <Vertragsart Typ="Laufzeit">
    <Tarif Name="Smart plus">19.7</Tarif>
  </Vertragsart>
</Beispielkosten>
```

Aufgabe 7:

Gegeben sei die folgende Datenbanktabelle mit Namen **TARIFE**.

ANBIETER	TARIF	NETZ	ANMERKUNG
Vodafone	CallYa Allnet	D2	prepaid
O2	Loop Smart L	o2	prepaid
Fonic	Fonic Smart	o2	500 Freieinheiten
freenetmobile	freeflat	D1	24 Monate Laufzeit

Füllen Sie die Lücken in der SQL/XML-Abfrage auf der **TARIFE**-Tabelle, die als Ergebnis die unten gezeigte Ausgabe liefert.

```
SELECT XMLELEMENT(
         NAME "Netz",
         XMLATTRIBUTES(__NETZ AS "Netzbetreiber"__),
         XMLAGG(
           XMLELEMENT(
             ___NAME "Tarif"___,
             XMLATTRIBUTES(ANBIETER AS "Anbieter"),
             ___TARIF___
           )
         )
       )
FROM TARIFE
GROUP BY __NETZ__
ORDER BY NETZ;
```

Ausgabe:

```
<Netz Netzbetreiber="D1">
  <Tarif Anbieter="freenetmobile">freeflat</Tarif>
</Netz>
<Netz Netzbetreiber="D2">
  <Tarif Anbieter="Vodafone">CallYa Allnet</Tarif>
</Netz>
<Netz Netzbetreiber="o2">
  <Tarif Anbieter="O2">Loop Smart L</Tarif>
  <Tarif Anbieter="Fonic">Fonic Smart</Tarif>
</Netz>
```

Aufgabe 8:

In unserer Sammlung animierter Verkehrszeichen betrachten wir heute das Gefahrenzeichen 128 *Bewegliche Brücke*.

(a) Ergänzen Sie die Lücken durch Angabe der passenden Ziffer! Hinweis: Nicht alle Ergänzungen werden gebraucht, manche aber mehrfach.

1 rect
2 repeatCount
3 fill
4 svg
5 path
6 points
7 line

```
<?xml version="1.0"?>
<___4___ xmlns="http://www.w3.org/2000/svg">
<rect x="0" y="0" width="600" height="600"
  fill="white"/>

<___7___ x1="50" y1="500" x2="550" y2="500"
  style="stroke:red; stroke-width:50px;
  stroke-linecap:round"/>
<___7___ x1="50" y1="500" x2="300" y2="67"
  style="stroke:red; stroke-width:50px;
  stroke-linecap:round"/>
<___7___ x1="300" y1="67" x2="550" y2="500"
  style="stroke:red; stroke-width:50px;
  stroke-linecap:round"/>
<path style="stroke:black; stroke-width:2px" fill="none"
  d="M 260 50 A 43 43 60 0 1 340 50 L 590 480
    A 43 43 -60 0 1 550 545 L 50 545
    A 43 43 -60 0 1 10 480 L 260 50"/>

<polygon ___6__="110,465 168,368 220,368 220,465"
  style="fill:black; stroke:none"/>
```

```
<polygon __6__="380,465 380,368 432,368 486,465"
  style="fill:black; stroke:none"/>

<__1__ x="220" y="348" width="160" height="20" fill="black">
  <animateTransform attributeName="transform" type="rotate"
    values="0 220 368; -40 220 368; 0 220 368" dur="5s"
    ___2__="indefinite"/>
</__1__>

<path style="stroke:black; stroke-width:15px" fill="none"
  d="M 225,420 q 16,-10 32,0 t 32,0"/>
<path style="stroke:black; stroke-width:15px" fill="none"
  d="M 288,421 q 16,-10 32,0 t 32,0"/>
<path style="stroke:black; stroke-width:15px" fill="none"
  d="M 225,450 q 16,-10 32,0 t 32,0"/>
<path style="stroke:black; stroke-width:15px" fill="none"
  d="M 288,451 q 16,-10 32,0 t 32,0"/>
<path style="stroke:black; stroke-width:15px" fill="none"
  d="M 351,422 q 16,-10 32,0 t 32,0"/>
<path style="stroke:black; stroke-width:15px" fill="none"
  d="M 351,452 q 16,-10 32,0 t 32,0"/>
<___7___ x1="225" y1="380" x2="225" y2="470"
  style="stroke:white; stroke-width:10px"/>
<___7___ x1="375" y1="380" x2="375" y2="470"
  style="stroke:white; stroke-width:10px"/>
</___4___>
```

(b) Wie wäre das **values**-Attribut im **animateTransform**-Element zu ändern, damit der Ankerpunkt (Drehpunkt) der Brücke rechts unten liegt, die Brücke aber gleich hoch öffnet?

values="___0 380 368; 40 380 368; 0 380 368___"

ENDE DER KLAUSUR