Gruppe A	PRÜFUNG AUS "SEMISTRUKTURIERTE DATEN" **184.705**		7. 1. 2014
Kennnr.	Matrikelnr.	Familienname	Vorname

Arbeitszeit: 100 Minuten. Aufgaben sind auf den Angabeblättern zu lösen; Zusatzblätter werden nicht gewertet.
Bitte tragen Sie Namen, Studienkennzahl und Matrikelnummer ein. Halten Sie Ihren Studentenausweis bereit.

Aufgabe 1: (12)

Betrachten Sie die folgende DTD Datei **test.dtd**:

```
<!ELEMENT A ((A,C)* | B+ | C+)>
<!ELEMENT B (C,D)*>
<!ELEMENT C (#PCDATA | B | A)*>
<!ELEMENT D (#PCDATA)>
<!ATTLIST B id ID #REQUIRED>
```

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

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

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

1. `<A><C><B>text</B></C><C>text</C></A>` gültig ◯ ungültig ⊗
2. `<A><C><B id="x01">text</B></C><C>text</C></A>` gültig ◯ ungültig ⊗
3. `<A><C><B id="x01"><C><B>text</B></C><D>text</D></B></C></A>` gültig ◯ ungültig ⊗
4. `<A><A><C>text</C></A><C><B id="x01"><C>text</C><D>text</D></B></C></A>` gültig ⊗ ungültig ◯
5. `<A><B id="x01"><C>text</C></B><B><C>text</C></B></A>` gültig ◯ ungültig ⊗
6. `<A><C>text</C><C>text</C><C>text</C></A>` gültig ⊗ ungültig ◯
7. `<A><B id="x01"><C>text</C></B><D>text</D></A>` gültig ◯ ungültig ⊗
8. `<A><B id="x01"><C>text</C><D>text</D></B></A>` gültig ⊗ ungültig ◯

(Pro korrekter Antwort 1.5 Punkte, **pro inkorrekter Antwort -1.5 Punkte**, pro nicht beantworteter Frage 0 Punkte, für die gesamte Aufgabe mindestens 0 Punkte)

Aufgabe 2: (15)

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

1. XML steht für eXtended Modeling Language — wahr ◯ falsch ⊗
2. Eine Eigenschaft von XML ist die Trennung von Struktur und Präsentation — wahr ⊗ falsch ◯
3. Die Reihenfolge von Attributen ist entscheidend — wahr ◯ falsch ⊗
4. Elemente ohne Namespace Präfix sind immer im leeren Namespace — wahr ◯ falsch ⊗
5. DTD bieten keine Unterstützung von Namespaces — wahr ⊗ falsch ◯
6. In einem XML-Schema spezifiziert xsd:sequence die Reihenfolge des Auftretens im Instanzdokument — wahr ⊗ falsch ◯
7. Die Verwendung von complexType mit simpleContent macht keinen Sinn — wahr ◯ falsch ⊗
8. Ein XML Parser prüft immer die Gültigkeit eines XML Dokuments — wahr ◯ falsch ⊗
9. SAX Filter können nicht verschachtelt verwendet werden — wahr ◯ falsch ⊗
10. Ein Absoluter XPath Pfad beginnt beim akutuellen Context Node — wahr ◯ falsch ⊗

(Pro korrekter Antwort 1.5 Punkte, **pro inkorrekter Antwort -1.5 Punkte**, pro nicht beantworteter Frage 0 Punkte, für die gesamte Aufgabe mindestens 0 Punkte)

Die folgenden Aufgaben 3 – 7 beziehen sich auf das XML-Dokument movies.xml, das Sie auf der letzten Seite dieser Prüfungsangabe finden.

Aufgabe 3: (12)

Vervollständigen Sie die XML Schema Datei **movies.xsd**, sodass XML-Dokumente in der Gestalt von **movies.xml** (siehe Anhang) bezüglich dieses Schemas gültig sind. XML-Dokumente der Form **movies.xml** sollen eine Liste von Filmen abspeichern. Berücksichtigen Sie beim Erstellen des Schemas folgende Punkte:

- Das Element `movies` ist das Wurzelelement und besteht aus mindestens zwei `movie`-Elementen.
- Jedes `movie`-Element beinhaltet Information zu einem Film. Ein Film besteht aus genau einem `title`-Element, genau einem `releaseYear`-Element und genau einem `imdbRating`-Element gefolgt von genau einem `characters`-Element. Die Reihenfolge dieser Elemente soll eingehalten werden.
- Das `characters`-Element hat mindestens zwei `name`-Elemente.
- Das `imdbRating`-Element darf einen Wert von 0 bis 10 beinhalten mit einer Nachkommastelle.
- Das `releaseYear`-Element darf einen Wert nicht kleiner als 1900 und nicht größer als 2014 beinhalten.
- Wählen Sie anhand des Dokuments im Anhang sinnvolle Typen und Häufigkeiten aus.
- Alle Attribute sollen verpflichtend sein. Dies soll in Ihrer XML-Schema Definition explizit ablesbar sein. Es sind keine Schlüssel zu definieren.

Datei **movies.xsd**:

```
<!-- Sie haben auch noch auf der folgenden Seite Platz! -->

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

   <xsd:element name="movies">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="movie" maxOccurs="unbounded" minOccurs="2" type="movie"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>

   <xsd:complexType name="movie">
      <xsd:sequence>
         <xsd:element name="title" type="xsd:string"/>
         <xsd:element name="releaseYear" type="releaseYearType"/>
         <xsd:element name="imdbRating" type="imdbRatingType"/>
         <xsd:element name="characters" type="charactersType"/>
      </xsd:sequence>
      <xsd:attribute name="url" type="xsd:string" use="required"/>
   </xsd:complexType>

   <xsd:simpleType name="releaseYearType">
      <xsd:restriction base="xsd:integer">
         <xsd:minInclusive value="1900"/>
         <xsd:maxInclusive value="2014"/>
      </xsd:restriction>
   </xsd:simpleType>
```

Datei **movies.xsd** (Fortsetzung):

```
   <xsd:simpleType name="imdbRatingType">
      <xsd:restriction base="xsd:decimal">
         <xsd:minInclusive value="0"/>
         <xsd:maxInclusive value="10"/>
         <xsd:fractionDigits value="1"/>
      </xsd:restriction>
   </xsd:simpleType>

   <xsd:complexType name="charactersType">
      <xsd:sequence>
         <xsd:element name="name" type="xsd:string" minOccurs="2" maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>

</xsd:schema>
```

Aufgabe 4: (10)

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

- Falls als Ergebnis mehrere Knoten selektiert werden, trennen Sie die jeweiligen Ausgaben durch einen senkrechten Strich.
- Falls als Ergebnis Elemente selektiert werden, geben Sie deren Inhalt an.
- Falls der angegebene XPath Ausdruck keine Knoten selektiert, notieren Sie im entsprechenden Feld "leere Ausgabe".

Betrachten Sie dazu folgendes Beispiel:

```
//movie/releaseYear
```

```
1994 | 1972 | 1974 | 1994
```

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

```
//movie[0]/title
```

```
leere Ausgabe
```

```
//movie[not(releaseYear/text() > 1990)]/title
```

```
Der Pate |  Der Pate2
```

```
//movie[characters/name = "Al Pacino"]/imdbRating
```

```
9.2 | 9.0
```

```
//characters/name[1]
```

```
Tim Robbins | Marlon Brando | Al Pacino | John Travolta
```

```
(//name)[6]
```

```
Robert De Niro
```

Aufgabe 5: (8)

Betrachten Sie folgende-XQuery Abfrage **movies.xq** angewandt auf **movies.xml**:

```
for $m in //movie
let $t := $m/title/text()
let $r := $m/releaseYear/text()
let $i := $m/imdbRating/text()
let $c := count($m//name)
where $i > 9
order by $r ascending
return <movie>{$t}({$r}) - {$i} - {$c}</movie>
```

Geben Sie nun die Ausgabe von **movies.xq** angewandt auf **movies.xml** an.
Die exakte Behandlung von Whitespaces ist dabei nicht relevant.

```
<movie>Der Pate(1972) - 9.2 - 2</movie>
<movie>Die Verurteilten(1994) - 9.2 - 2</movie>
```

Aufgabe 6: (10)

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

- Es soll ein Element `movies` erzeugt werden, das folgende Elemente enthält.
- Es soll ein Element `number` erzeugt werden, das die Anzahl der `movie`-Elemente enthält.
- Es soll ein Element `characters` erzeugt werden, das die Anzahl der `name`-Elemente enthält.
- Es soll ein Element `maxRating` erzeugt werden, welches das höchste Rating aller `imdbRating`-Elemente enthält.
- Es soll ein Element `minRating` erzeugt werden, welches das niedrigste Rating aller `imdbRating`-Elemente enthält.
- Es soll ein Element `urls` erzeugt werden, welches alle `url`-Attribute aller `movie`-Elemente in `u`-Elementen kapselt.

Für das Dokument **movies.xml** soll beispielsweise folgende Ausgabe erzeugt werden:

```
<movies>
   <number>4</number>
   <characters>8</characters>
   <maxRating>9.2</maxRating>
   <minRating>8.9</minRating>
   <urls>
      <u>http://www.imdb.com/?movie=...</u>
      <u>http://www.imdb.com/?movie=...</u>
      <u>http://www.imdb.com/?movie=...</u>
      <u>http://www.imdb.com/?movie=...</u>
   </urls>
</movies>
```

Vervollständigen Sie hier das XSLT-Stylesheet **movies.xsl**. Die Verwendung von Kontrollstrukturen wie `xsl:for-each`, `xsl:if`, etc. ist für die Lösung *nicht* erlaubt! Sie brauchen sich nicht um Whitespaces etc. zu kümmern.

Datei **movies.xsl**:

```
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

   <xsl:template match="/">
      <movies>
         <number><xsl:value-of select="count(//movie)"/></number>
         <characters><xsl:value-of select="count(//name)"/></characters>
         <maxRating><xsl:value-of select="max(//imdbRating)"/></maxRating>
         <minRating><xsl:value-of select="min(//imdbRating)"/></minRating>
         <urls>
           <xsl:apply-templates select="//movie"/>
         </urls>
      </movies>
   </xsl:template>

   <xsl:template match="movie">
      <u><xsl:value-of select="@url"/></u>
   </xsl:template>

</xsl:stylesheet>
```

Aufgabe 7: (8)

Vervollständigen Sie den folgenden SAX Handler, der angewandt auf Dokumente der Gestalt **movies.xml** den Durchschnittswert aller `imdbRating`-Elemente ermittelt und in einer Variable `avg` speichert.

Für das Dokument **movies.xml** wäre der Durchschnittswert beispielsweise 9.075. [(9.2 + 9.2 + 9.0 + 8.9)/4]

```
public class SSDHandler extends DefaultHandler {

   float avg = 0;
   float sum = 0;
   float current = 0;
   int nr = 0;
   boolean inImdbRating = false;
   StringBuffer sb = new StringBuffer();

   public void startElement(String uri, String localName, String name,
         Attributes attributes) throws SAXException {
      if("imdbRating".equals(localName)){
         inImdbRating = true;
         nr++;
      }
   }

   public void endElement(String uri, String localName, String name)
         throws SAXException {
      if(inImdbRating){
         sum += current;
         inImdbRating = false;
         sb.delete(0, sb.length());

         avg = sum / nr;
      }
   }

   public void characters(char[] ch, int start, int length) throws SAXException {
      if(inImdbRating){
         sb.append(ch, start, length);
         current = Float.parseFloat(sb.toString());
      }
   }
}
```

Sie können diese Seite abtrennen!

Datei **movies.xml**:

```
<?xml version="1.0" encoding="UTF-8"?>

<movies>

   <movie url="http://www.imdb.com/?move=....">
      <title>Die Verurteilten</title>
      <releaseYear>1994</releaseYear>
      <imdbRating>9.2</imdbRating>
      <characters>
         <name>Tim Robbins</name>
         <name>Morgan Freeman</name>
      </characters>
   </movie>

   <movie url="http://www.imdb.com/?move=....">
      <title>Der Pate</title>
      <releaseYear>1972</releaseYear>
      <imdbRating>9.2</imdbRating>
      <characters>
         <name>Marlon Brando</name>
         <name>Al Pacino</name>
      </characters>
   </movie>

   <movie url="http://www.imdb.com/?move=....">
      <title>Der Pate 2</title>
      <releaseYear>1974</releaseYear>
      <imdbRating>9.0</imdbRating>
      <characters>
         <name>Al Pacino</name>
         <name>Robert De Niro</name>
      </characters>
   </movie>

   <movie url="http://www.imdb.com/?move=....">
      <title>Pulp Fiction</title>
      <releaseYear>1994</releaseYear>
      <imdbRating>8.9</imdbRating>
      <characters>
         <name>John Travolta</name>
         <name>Samuel L. Jackson</name>
      </characters>
   </movie>

</movies>
```

Gesamtpunkte: 75