PRÜFUNG IN "SEMI-STRUKTURIERTE DATEN" **184.705**			26. 11. 2018
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 folgende XML Schema Datei **test.xsd**:

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

    <xsd:element name="Y" type="Ytype"/>

    <xsd:complexType name="Ytype">
        <xsd:all>
            <xsd:element name="X" type="Xtype" minOccurs="0" maxOccurs="1"/>
            <xsd:element name="Y" type="Ytype" minOccurs="0" maxOccurs="1"/>
        </xsd:all>
    </xsd:complexType>

    <xsd:complexType name="Xtype">
        <xsd:choice minOccurs="0" maxOccurs="1">
            <xsd:element name="X" type="Xtype"/>
            <xsd:element name="C" type="xsd:int"/>
        </xsd:choice>
    </xsd:complexType>

</xsd:schema>
```

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.xsd zu entscheiden.

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

1. `<Y/>` valid ⊗ invalid ○
2. `<Y><X/><Y/></Y>` valid ⊗ invalid ○
3. `<Y><X><Y/></X></Y>` valid ○ invalid ⊗
4. `<Y><Y/><Y/></Y>` valid ○ invalid ⊗
5. `<Y><X><C>1</C><C>2</C></X></Y>` valid ○ invalid ⊗
6. `<Y><X><X><X/></X></X></Y>` valid ⊗ invalid ○
7. `<Y><X><C>1</C></X></Y>` valid ⊗ invalid ○
8. `<Y><X><C>SSD</C></X></Y>` valid ○ invalid ⊗

(Für jede korrekte Antwort 1.5 Punkte, **für jede falsche Antwort -1.5 Punkte**, unbeanwortete Fragen 0 Punkt, Insgesamt nicht weniger als 0 Punkte)

Aufgabe 2: (15)
Beantworten Sie, die folgenden Fragen kurz und bündig (Für jede korrekte Antwort 1.5 Punkte).

1. Erläutern Sie kurz die Einschränkungen von DTDs bei der Definition von Fremdschlüsseln.

   Antwort: Es kann nicht definiert auf welches Schlüsselattribut sich der Fremdschlüssel bezieht.

2. Welche Sprachen um Schemata zu definieren haben wir in der Vorlesung kennengelernt?

   Antwort: DTDs, XML Schema

3. Welche in der Vorlesung behandelte API hat nur konstanten Speicherbedarf?

   Antwort: SAX

4. Zu welchem Zweck werden Namespaces verwendet?

   Antwort: Um Elemente/Attribute mit gleichem Namen aus unterschiedlichen Quellen zu unterscheiden. Um Elemente/Attribute zu gruppieren.

5. Wie Unterscheiden sich Elementen von Attributen bezüglich der Signifikanz der Reihenfolge im XML Dokument?

   Antwort: Die Reihenfolge ist nur für Elemente nicht aber für Attribute signifikant.

6. Geben Sie drei der möglichen Achsen in XPath an.

   Antwort: ancestor, ancestor-or-self, attribute, child, descendant, descendant-or-self, following, following-sibling, namespace, parent, preceding, preceding-sibling, self

7. Was macht das Default-Template in XSLT für Attribute?

   Antwort: Es gibt den Wert des Attributs aus.

8. Welches Datenmodell verwenden wir für semistrukturierte Daten?

   Antwort: Bäume mit Labels auf den Kanten.

9. Wieso können URIs nicht direkt als prefix für XML Elemente/Attribute verwendet werden?

   Antwort: Die resultierenden Elementnamen sind keine gütigen XML-Namen.

10. Erläutern Sie kurz den Unterschied zwischen `for` und `let` statements in XQuery.

    Antwort: Mit for wird jedes Element einer Sequenz einzeln an die Variable gebunden (es wird über die Elemente der Sequenz iteriert). Mit let wird die vollständige Sequenz auf einmal an die Variable gebunden.

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

Aufgabe 3: (12)
Vervollständigen Sie das DTD Dokument **cinema.dtd**, sodass XML-Dokumente in der Gestalt von **cinema.xml** (siehe Anhang) bezüglich dieser DTD gültig sind. Berücksichtigen Sie beim Erstellen der DTD folgende Punkte:

- Das Wurzel Element hat drei Kind-Elemente `cinema_halls`, `movies` und `program`, wobei `cinema_halls` mindestens ein Kind-Element `hall` hat, und die anderen beliebig viele Kinder des entsprechenden Typ `movie` oder `screening` haben.
- Jedes `hall` Element enthält den Namen als Text Content. Zusätzlich wird ein `id` Attribut gespeichert das eine eindeutige Saalnummer definiert.
- Jedes `movie` Element enthält ein `title` Element, kann ein `director` Element enthalten und enthält beliebig viele `actor` Elemente (in dieser Reihenfolge). Zusätzlich wird ein `nr` Attribut gespeichert das jedem Film eine eindeutige Nummer zuordnet.
- Jedes `screening` Element hat beliebig viele `play` Elemente als Kinder. Zusätzlich wird das Datum als `date` Attribut gespeichert und ein Verweis auf den vorgeführten Film im `movie` Attribut.
- `play` Elemente sind immer leer und haben zwei Attribute. Das Attribut `time` speichert die Vorführzeit und das Attribut `hall` verweist auf den Kinosaal in dem der Film gezeigt wird.
- Wenn nicht angegeben treffen Sie plausible Annahmen über Typen von Attributen und Elementen.

File **cinema.dtd**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT cinema (cinema_halls, movies, program)>
<!ELEMENT cinema_halls (hall)+>
<!ELEMENT hall (#PCDATA)>
<!ATTLIST hall id ID #REQUIRED>
<!ELEMENT movies (movie)*>
<!ELEMENT movie (title, director?, actor*)>
<!ATTLIST movie nr ID #REQUIRED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT director (#PCDATA)>
<!ELEMENT actor (#PCDATA)>
<!ELEMENT program (screening)*>
<!ELEMENT screening (play)*>
<!ATTLIST screening date CDATA #REQUIRED>
<!ATTLIST screening movie IDREF #REQUIRED>
<!ELEMENT play (EMPTY)>
<!ATTLIST play time CDATA #REQUIRED>
<!ATTLIST play hall IDREF #REQUIRED>
```

Aufgabe 4: (10)

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

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

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

```
count(//screening[play[@hall="s3"]])
```

```
2
```

```
//hall[@id=//screening[@movie="f3"]/play/@hall]/text()
```

```
Audi Max
Saal 4
```

```
//movies/movie[1]/actor[2]
```

```
<actor>Al Pacino</actor>
```

```
//movies/movie[@nr=//screening/@movie][last()]/title/text()
```

```
Star Wars - A New Hope
```

```
//movie[not(director)][@nr=//screening/@movie]/actor[2]
```

```
<actor>Mark Hamill</actor>
```

Aufgabe 5: (8)

Betrachten Sie die folgende Java Klasse und geben Sie die Ausgabe der Klasse an, wenn als Input die Datei **cinema.xml** verwendet wird.

```
public class RunSAX extends DefaultHandler {
  private String eleText, id, date;
  private HashMap<String, List<String>> s = new HashMap<String, List<String>>();
  private HashMap<String, String> h = new HashMap<String, String>();

  public void characters(char[] text, int start, int length)
    throws SAXException {
    eleText = new String(text, start, length);
  }

  public void startElement(String namespaceURI, String localName, String qName, Attributes atts)
    throws SAXException {
    if ("hall".equals(localName))
      id = atts.getValue("id");

    if ("screening".equals(localName))
      date = atts.getValue("date");

    if ("play".equals(localName)) {
      String v = date + "␣um␣" + atts.getValue("time");
      id = atts.getValue("hall");
      if(!s.containsKey(id))
        s.put(id, new LinkedList<String>());
      s.get(id).add(v);
    }
  }

  public void endElement(String namespaceURI, String localName, String qName)
    throws SAXException {
    if ("hall".equals(localName))
      h.put(id, eleText);
  }

  public void endDocument() throws SAXException {
    for (String hk : h.keySet()) {
      System.out.println(h.get(hk));
      Collections.sort(s.get(hk));
      for (String t : s.get(hk)) {
        System.out.println(t);
      }
    }
  }

  public static void main(String[] args) throws Exception {
    InputSource source = new InputSource(new FileInputStream("cinema.xml"));

    XMLReader xr = XMLReaderFactory.createXMLReader();
    RunSAX rs = new RunSAX();
    xr.setContentHandler(rs);

    xr.parse(source);
  }
}
```

Geben Sie hier die Ausgabe an:

```
Saal 3
2018-10-26 um 20:45
2019-10-26 um 20:15
Saal 4
2018-10-26 um 22:30
Audi Max
2018-10-26 um 19:15
2018-10-26 um 22:00
Saal 2
2018-10-26 um 21:00
2018-10-26 um 22:00
```

(Reihenfolge undefiniert)

Aufgabe 6: (10)

Schreiben Sie eine XQuery Abfrage um folgende Aufgabenstellung zu lösen:

> Es soll ein XML Dokument mit der Wurzel `report` erzeugt werden.
>
> In diesem Dokument sollen alle Filme ausgegeben werden, in denen "Elijah Wood" nicht spielt (es gibt im entsprechenden `movie` Node kein `actor` Kind mit dem Inhalt "Elijah Wood").
>
> Für jeden Film wird der Titel in einem `title` Element ausgegeben. Weiters gibt es für jeden Tag an dem der Film gezeigt wird ein `shown` Element. Das Datum sowie die Anzahl der Vorführungen wird in Elementen `date` und `amount` wird im `shown` Element ausgegeben.

Ihre Abfrage soll, angewandt auf **cinema.xml**, folgende Ausgabe liefern (Die Ordnung der Filme darf abweichen):

```
<report>
  <movie>
    <shown>
      <date>2018-10-26</date>
      <amount>4</amount>
    </shown>
    <title>The Godfather</title>
  </movie>
  <movie>
    <shown>
      <date>2018-10-26</date>
      <amount>2</amount>
    </shown>
    <shown>
      <date>2018-10-27</date>
      <amount>0</amount>
    </shown>
    <title>Star Wars - A New Hope</title>
  </movie>
</report>
```

Geben Sie nun die XQuery Abfrage an:

```
<report>
{
  for $m in doc("cinema.xml")//movie
  where every $a in $m/actor satisfies $a != "Elijah Wood"
  return <movie>{
    for $s in doc("cinema.xml")//screening[@movie=$m/@nr]
    return
    <shown>
      <date>{data($s/@date)}</date>
      <amount>{count($s/play)}</amount>
    </shown>
  }
  <title>{$m/title/text()}</title>
  </movie>
}
</report>
```

Aufgabe 7: (8)

Betrachten Sie folgendes XSLT Dokument **cinema.xsl**:

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

 <xsl:template match="cinema">
  <xsl:apply-templates select="cinema_halls" />
 </xsl:template>

 <xsl:template match="cinema_halls">
  <xsl:for-each select="hall">
    Films in hall <xsl:value-of select="current()"/>  :
   <xsl:variable name="hallid" select="current()/@id" />

   <xsl:for-each select="/cinema//movie[/cinema//screening[play/@hall=$hallid]/@movie=./@nr]">
    <xsl:apply-templates select="." />
    <xsl:variable name="movid" select="./@nr" />
    <xsl:for-each select="/cinema//screening[play/@hall=$hallid␣and␣@movie␣=$movid]" >
      <xsl:value-of select="@date"/> : <xsl:apply-templates select="play[@hall=$hallid]" />
    </xsl:for-each>
   </xsl:for-each>

  </xsl:for-each>
 </xsl:template>

 <xsl:template match="movie">
   <xsl:value-of select="title"/>
   from  <xsl:value-of select="director"/> <xsl:if test="not(director)">Anon.</xsl:if>
 </xsl:template>

 <xsl:template match="play">
   <xsl:value-of select="@time"/> <xsl:if test="following-sibling::play[@hall=current()/@hall]">, </xsl:if>
 </xsl:template>

</xsl:stylesheet>
```

Geben Sie nun die Ausgabe von **cinema.xsl** angewandt auf **cinema.xml** an.

Sie müssen sich nicht um Whitespaces kümmern.

```
Films in hall Audi Max:
The Godfather
from  Mario Puzo
2018-10-26 : 19:15
Star Wars - A New Hope
from  Anon.
2018-10-26 : 22:00

Films in hall Saal 2:
The Godfather
from  Mario Puzo
2018-10-26 : 21:00, 22:00

Films in hall Saal 3:
The Godfather
from  Mario Puzo
2018-10-26 : 20:45
The Lord of the Rings: The Fellowship of the Ring
from  Peter Jackson
2019-10-26 : 20:15

Films in hall Saal 4:
Star Wars - A New Hope
from  Anon.
2018-10-26 : 22:30
```

Total points: 75

File **cinema.xml**: **You may separate this page!**

```
<?xml version="1.0" encoding="UTF-8"?>
<cinema>
  <cinema_halls>
    <hall id="am">Audi Max</hall>
    <hall id="s2">Saal 2</hall>
    <hall id="s3">Saal 3</hall>
    <hall id="s4">Saal 4</hall>
  </cinema_halls>
  <movies>
    <movie nr="f1">
      <title>The Godfather</title>
      <director>Mario Puzo</director>
      <actor>Marlon Brando</actor>
      <actor>Al Pacino</actor>
    </movie>
    <movie nr="f2">
      <title>The Lord of the Rings: The Fellowship of the Ring</title>
      <director>Peter Jackson</director>
      <actor>Elijah Wood</actor>
      <actor>Ian McKellen</actor>
    </movie>
    <movie nr="f3">
      <title>Star Wars - A New Hope</title>
      <actor>Harrison Ford</actor>
      <actor>Mark Hamill</actor>
    </movie>
  </movies>
  <program>
    <screening date="2018-10-26" movie="f3">
      <play time="22:00" hall="am"/>
      <play time="22:30" hall="s4"/>
    </screening>
    <screening date="2018-10-26" movie="f1">
      <play time="19:15" hall="am"/>
      <play time="21:00" hall="s2"/>
       <play time="22:00" hall="s2"/>
      <play time="20:45" hall="s3"/>
    </screening>
    <screening date="2019-10-26" movie="f2">
      <play time="20:15" hall="s3"/>
    </screening>
    <screening date="2018-10-27" movie="f3"/>
  </program>
</cinema>
```