PRÜFUNG IN "SEMI-STRUKTURIERTE DATEN" **184.705**			23.06.2020
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**:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="B">
        <xsd:complexType mixed="true" >
            <xsd:choice maxOccurs="2">
                <xsd:element name="C" minOccurs="0" type="xsd:integer"/>
                <xsd:element name="A" type="typeA"/>
            </xsd:choice>
        </xsd:complexType>
        <xsd:key name="key1">
            <xsd:selector xpath=".//A"/>
            <xsd:field xpath="@id"/>
         </xsd:key>
    </xsd:element>

    <xsd:complexType name="typeA" mixed="false">
         <xsd:sequence>
                <xsd:element name="A" type="typeA" minOccurs="0" maxOccurs="2"/>
          </xsd:sequence>
          <xsd:attribute name="id" type="xsd:integer"/>
    </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. `<B>SSD</B>` valid ○ invalid ○
2. `<B>DBAI<C>1842</C></B>` valid ○ invalid ○
3. `<B><A/></B>` valid ○ invalid ○
4. `<B><A id="1"><A id="4"><A id="1"></A></A></A></B>` valid ○ invalid ○
5. `<B><A id="3"><A id="1"><A id="4"></A></A></A></B>` valid ○ invalid ○
6. `<B><C>42</C><A id="1"/></B>` valid ○ invalid ○
7. `<B><C>42</C><A id="1"/><C>23</C></B>` valid ○ invalid ○
8. `<A></A>` 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. Welche Sprachen um Schemata zu definieren haben wir in der Vorlesung kennengelernt?

2. Zu welchem Zweck werden Namespaces verwendet?

3. Wie Unterscheiden sich Elemente von Attributen bezüglich der Signifikanz der Reihenfolge im XML Dokument?

4. Welche Art von Parsern, neben Tree-based Parsern, haben wir in der VO noch kennengelernt?

5. Was macht das Default-Template in XSLT für Elemente?

6. Wie geht XSLT mit der Situation um wenn mehrere Templates auf ein Element matchen?

7. Beschreiben Sie kurz den Unterschied zwischen wohl-geformten (well-formed) XML-Dokumenten und validen XML Dokumenten.

8. Erläutern Sie kurz die Einschränkungen von DTDs bei der Definition von mehreren Schlüsseln für ein Dokument.

9. Geben Sie einen der signifikanten Unterschiede zwischen HTML und XML an.

10. Welche in der Vorlesung behandelte API erlaubt wahlfreien Zugriff auf das gesamte XML Dokument?

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

Aufgabe 3: (12)

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

- Ein `exhibition` Element enthält genau ein `name` Element, gefolgt von mindestens einem `art-piece` Element (möglicherweise mehrere). Jedes `exhibition` Element muss Werte für die Attribute `starts` und `ends` enthalten.
- Ein `art-piece` Element enthält die folgenden Kinder in genau dieser Reihenfolge: Ein `title` Element, ein `description` Element, und optional ein `creation-time` Element.
- Ein `art-piece` Element *kann* ein `author` Attribut beinhalten, welches nur auf eine id im Dokument verweisen darf. Ein `art-piece` Element *muss* ein Attribut `room` beinhalten, welches auch nur auf eine id im Dokument verweisen darf.
- Ein `room` Element ist entweder leer oder enthält genau ein `name` Element. Ein `room` Element muss ein `id` Attribute enthalten welches im Dokument einzigartig sein muss.
- Ein `author` Element muss eine der folgenden Bedingungen erfüllen: (i) Es enthält genau ein `name` Element als Kind, oder (ii) es enthält ein oder mehrere `first` Elemente, gefolgt von genau einem `last` Element.
- Ein `author` Element kann ein `id` Attribut enthalten. Wenn ein `author` Element ein solches Attribut enthält muss der Wert des Attributes einzigartig im Dokument sein.
- Wenn nicht angegeben treffen Sie plausible Annahmen über Typen von Attributen und Elementen.

File **museum.dtd**:

```
<!ELEMENT museum (exhibition+,room+,author+)>
```

Aufgabe 4: (10)

Betrachten Sie die folgenden XPath-Abfragen angewandt auf das Dokument **museum.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.

```
//name[../self::author]
```

```
//art-piece[1]/title
```

```
//room[@id = //exhibition[1]//@room]/name
```

```
count(//*[../@id])
```

```
count(//art-piece[not(@author)])
```

Aufgabe 5: (10)

Ergänzen Sie die Java Klasse auf dieser und der nächsten Seite so, dass sie beim Aufruf, mit XML files die dem Schema der gegebenen **museum.xml** folgen, den folgenden Output erzeugt:

- Es ist gefragt, dass Sie die vollen Namen aller AutorInnen und, wenn vorhanden, die id der AutorIn ausgegeben wird. Beachten Sie dabei folgende Punkte:
  1. Es soll eine Liste aller AutorInnen ausgegeben werden. Dabei soll jeder Eintrag die Form "name (id)" haben (siehe Beispiel Output).
  2. Wenn der Name einer AutorIn mittels `first`/`last` Elementen angegeben wurde soll der volle Name als ein String erzeugt werden. Dabei kommen zuerst die Vornamen, in der selben Reihenfolge wie im XML Dokument und zuletzt der Nachname. **Vergessen Sie nicht darauf, dass es mehrere `first` Elemente im selben `author` Element geben kann.**
  3. Wenn ein `author` Element kein `id` Attribut hat soll stattdessen "No Id" in Klammer nach dem Namen ausgegeben werden (siehe Beispiel Output).
- Halten Sie sich bei der Formatierung an den Beispiel Output.

Beispiel Output beim Aufruf mit der gegebenen **museum.xml**:

```
Jan van Goyen (author1)
John Smith (No id)
Margareta Haverman (author2)
```

Vervollständigen Sie den folgenden Code:

```
public class RunSAX extends DefaultHandler {
  String eleText;









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 {









}
```

```
  public void endElement(String namespaceURI, String localName, String qName) throws SAXException {

  }
  public void endDocument() throws SAXException {

  }

  public static void main(String[] args) throws Exception {
    InputSource source = new InputSource(new FileInputStream("auction.xml"));
    XMLReader xr = XMLReaderFactory.createXMLReader();
    RunSAX rs = new RunSAX();
    xr.setContentHandler(rs);
    xr.parse(source);
  }
}
```

Aufgabe 6: (8)

Betrachten Sie folgende XQuery **museum.xq**:

```
<art>
{
  for $r in doc("museum.xml")//room
  let $rd := $r/@id
  where $r/name != ""
  return <r n="{$r/name}">
  {
    for $p in doc("museum.xml")//art-piece[@room=$rd]
    order by $p/parent::exhibition/@starts ascending
    return <p>{$p/title/text()}</p>
  }
  </r>
}
</art>
```

Geben Sie nun die Ausgabe von **museum.xq** angewandt auf **museum.xml** an.

Sie müssen sich nicht um Whitespaces kümmern.

Aufgabe 7: (10)

Betrachten Sie das folgende XSLT Dokument **museum.xsl**:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
 <xsl:output method="text"/>
    <xsl:template match="museum">
        Exhibitions:
        <xsl:apply-templates select="//exhibition[not( art-piece[@author = 'author1'])]"/>
    </xsl:template>

    <xsl:template match="exhibition">
        Exhibition: <xsl:value-of select="name"/>
        (From: <xsl:value-of select="./@starts"/>, To: <xsl:value-of select="current()/@ends"/>)

        # of Art-Pieces: <xsl:value-of select="count(./art-piece)"/>

        Rooms:
        <xsl:call-template name="room-printer">
            <xsl:with-param name="mode" select = "'exhibition'"/>
        </xsl:call-template>

        <xsl:apply-templates select="./art-piece"/>
    </xsl:template>

    <xsl:template match="art-piece">
        Art-Piece: <xsl:value-of select="title"/>
        Desc.: <xsl:value-of select="description"/>

        <xsl:call-template name="room-printer">
            <xsl:with-param name="mode" select = "artpiece"/>
            <xsl:with-param name="id" select = "current()/@room"/>
        </xsl:call-template>

        <xsl:variable name="var" select="current()/@author" />
        <xsl:value-of select="//author[@id = $var]/name | //author[@id = $var]/last"/>
    </xsl:template>

    <xsl:template match="room">
        <xsl:value-of select="name" /> (ID: <xsl:value-of select="./@id"/>)
    </xsl:template>

    <xsl:template name="room-printer">
        <xsl:param name = "mode"/> <xsl:param name = "id"/>

        <xsl:choose>
            <xsl:when test="$mode = 'exhibition'">
                <xsl:for-each select="//room[@id = current()/art-piece/@room]">
                    <xsl:sort select="current()/@id"/>
                    Room: <xsl:apply-templates select="current()"/>   .
                </xsl:for-each>
            </xsl:when>
            <xsl:otherwise>
                Room: <xsl:apply-templates select="//room[@id = $id]"/>
            </xsl:otherwise>
        </xsl:choose>
    </xsl:template>
</xsl:stylesheet>
```

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

Sie müssen sich nicht um Whitespaces kümmern. Sie dürfen lange Strings abkürzen, solange **eindeutig** ist auf was verwiesen wird, z.B.:“A scene of . . .” statt “A scene of fishermen casting their net in front of a fortress”.

Datei **museum.xml**: **Sie können diese Seite abtrennen!**

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE museum SYSTEM "museum.dtd">
<museum>
    <exhibition starts="04.04.2020" ends="09.09.2020">
      <name>Dutch Masterpieces</name>

      <art-piece author="author1" room = "room1">
          <title>Castle by a River</title>
          <description>A scene of fishermen casting their net in front of a fortress</description>
          <creation-time>1647</creation-time>
      </art-piece>

      <art-piece author="author2" room = "room3">
          <title>A Vase of Flowers</title>
          <description>The artist's skill is on full display in this magnificent
            arrangement of flowers and fruit.</description>
          <creation-time>1716</creation-time>
      </art-piece>
    </exhibition>

    <exhibition starts="01.01.2020" ends="03.04.2020">
      <name>The Art of Native America</name>

      <art-piece room = "room3">
          <title>Snow goggles</title>
          <description>A hunter used these goggles to shield his eyes
            from the wind and the sun.</description>
          <creation-time>ca. 800-1200</creation-time>
      </art-piece>

      <art-piece room = "room1">
          <title>Soul catcher</title>
          <description>A shaman employed this sculpture to recover
            the lost souls of his patients.</description>
          <creation-time>ca. 1840-60</creation-time>
      </art-piece>
    </exhibition>

    <room id="room1">
      <name>The Main Hall</name>
    </room>
    <room id="room2">
      <name>The Knights' Room</name>
    </room>
    <room id="room3">
      <name>The Rembrandt Room</name>
    </room>
    <room id="room4"/>

    <author id="author1">
      <name>Jan van Goyen</name>
    </author>
    <author id="author2">
      <first>Margareta</first>
      <last>Haverman</last>
    </author>
    <author>
      <name>John Smith</name>
    </author>
</museum>
```