PRÜFUNG IN "SEMI-STRUKTURIERTE DATEN" **184.705**			28. 10. 2019
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="W">
  <xsd:complexType>
  <xsd:sequence>
    <xsd:element name="A" minOccurs="0" type="xsd:boolean"/>
    <xsd:element name="B" maxOccurs="3" type="typeB"/>
  </xsd:sequence>
  </xsd:complexType>
  <xsd:unique name="key1">
      <xsd:selector xpath=".//B"/>
      <xsd:field xpath="@svnr"/>
  </xsd:unique>
</xsd:element>
<xsd:complexType name="typeB" mixed="true">
  <xsd:sequence>
      <xsd:element name="C" type="typeB" minOccurs="0" maxOccurs="2"/>
  </xsd:sequence>
  <xsd:attribute name="svnr" type="xsd:integer"/>
</xsd:complexType>
</xsd:schema>
```

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

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

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

1. `<W><A>1</A><B svnr="1111222222">Tim</B></W>` valid ⊗ invalid ○
2. `<W><B>Jill</B></W>` valid ⊗ invalid ○
3. `<W><B svnr="1111222222">Randy</B><B svnr="1111222222">Mark</B></W>` valid ○ invalid ⊗
4. `<W><B svnr="1111222222">Wilson <C svnr="1111222222">Al</C></B></W>` valid ⊗ invalid ○
5. `<W><A>0</A><A>1</A><B/></W>` valid ○ invalid ⊗
6. `<W><A>Falsch</A><B svnr="1111222222">Brad <C/></B></W>` valid ○ invalid ⊗

(Für jede korrekte Antwort 2 Punkte, **für jede falsche Antwort -2 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 werden auf welches Schlüsselattribut sich der Fremdschlüssel bezieht.

2. Zu welchem Zweck werden Namespaces verwendet?

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

3. 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

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

   Antwort: Event-based Parser, SAX.

5. Was versteht man unter FLWOR Expressions?

   Antwort: Die typische Struktur einer XQuery: For Let Where Order by Return

6. Geben Sie ein Beispiel für eine Document Type Declaration.

   Antwort:

7. Definieren sie kurz die vier verschiedenen Arten von Inhalt (content) die ein xml-Element haben kann. (6)

   Antwort:

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

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

- Das Wurzel Element enthält `conference_paper`, `article` und `reg_author` Elemente in beliebiger Reihenfolge und Häufigkeit.
- `conference_paper` Elemente enthalten eine textuelle Beschreibung mit jeweils einem `title`, `conference` Element und einem oder mehreren `author` Elementen. `author` Elemente könnten ein Attribut `ref` haben das auf ein `reg_author` Element referenziert.
- `reg_author` Elemente enthalten entweder ein `name` Element oder ein `first_name` Element gefolgt von einem `last_name` Element. Jedes `reg_author` Element hat ein Attribut mit einer eindeutigen ID.
- `article` Elemente enthalten ein `title` Element ein `journal` oder `type` Element und mindestens ein `author` Element. Das `type` Element hat ein Attribut `printed` das die Werte `yes` und `no` annehmen kann. Wird das Attribut im Dokument nicht aufgeführt soll der Wert `yes` angenommen werden.
- Wenn nicht angegeben treffen Sie plausible Annahmen über Typen von Attributen und Elementen.

File **bibliography.dtd**:

```
<!ELEMENT bibliography (reg_author|conference_paper|article)*>

<!ELEMENT conference_paper (#PCDATA | title | conference | author)*>

<!ELEMENT title (#PCDATA)>
<!ELEMENT conference (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ATTLIST author ref IDREF #IMPLIED>

<!ELEMENT reg_author (name|(first_name,last_name))>
<!ATTLIST reg_author id ID #REQUIRED>

<!ELEMENT name (#PCDATA)>
<!ELEMENT first_name (#PCDATA)>
<!ELEMENT last_name (#PCDATA)>

<!ELEMENT article (title,(journal|type),author+)>

<!ELEMENT journal (#PCDATA)>
<!ELEMENT type (#PCDATA)>
<!ATTLIST type printed (yes|no)  "yes">
```

Aufgabe 4: (10)

Betrachten Sie die folgenden XPath-Abfragen angewandt auf das Dokument **bibliography.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(//.[./author[@ref]])
```

```
4
```

```
//author[@ref="a1"]/../author[not(@ref)]
```

```
<author>G. Gottlob</author>

<author>R. Pichler</author>
```

```
//article[journal][2]/title
```

```
leere Ausgabe
```

```
//article[2][type[@printed="yes"]]/title
```

```
<title>SSD 2019-6 exam</title>
```

```
//reg_author/name/../@id
```

```
a3
```

Aufgabe 5: (8)

Betrachten Sie die folgende Methode run wie unten gegeben. Geben Sie nun die Ausgabe dieser Methode an. Nehmen Sie dabei an, dass die Variable doc die DOM Repräsentation des XML Dokuments **bibliography.xml** enthält.

Hinweis: Die Methode trim() entfernt alle Whitespaces an Anfang und Ende des Strings.

```
public static void run(Document doc) throws Exception {
    NodeList nodes_conf = doc.getElementsByTagName("conference");

    for(int i=0; i < nodes_conf.getLength(); i++) {
        Node tnode = nodes_conf.item(i);
        String text = tnode.getTextContent();

        Node next_sib = tnode.getNextSibling();
        int c = 0;
        while(next_sib != null) {
            if(next_sib.getNodeName().equals("author"))
                c = c + 1;
            next_sib = next_sib.getNextSibling();
        }

        tnode = tnode.getPreviousSibling();
        while(tnode != null && tnode.getTextContent().trim().isEmpty())
            tnode = tnode.getPreviousSibling();
        String a  = tnode.getTextContent();
        System.out.println(text + ",␣" + a + ",␣" + c);
    }
}
```

```
AMW 2019,  was published at, 3
AMW 2019, Semantic Width Revisited, 3
AAAI 2019, Complexity of Abstract Argumentation under a Claim-Centric View., 3
```

Aufgabe 6: (10)
Schreiben Sie eine XQuery Abfrage um folgende Aufgabenstellung zu lösen:

Es soll ein XML Dokument mit der Wurzel `papers` erzeugt werden.

Für jedes `title` Element der Bibliographie gibt es ein `p` Element direkt unter der Wurzel mit dem attribut `name`, dass den Titel als String enthält. **Ausnahme: Falls der `title` den text "exam" enthält soll dafür kein `p` Element erzeugt werden.**

In jedem `p` Element soll es zwei Unterelemente geben `is_article` und `is_conf`. Dabei enthält das `is_article` Element den Inhalt "yes" falls es einen Artikel mit dem Titel in der Bibliographie gibt, sonst "no". Analog ist der Inhalt für `is_conf` "yes" falls es ein Konferenzpaper zum Titel gibt, sonst "no".

Hinweis: Ob eine string s den text "xyz" enthält können sie in XQuery mittels `contains(s, ''xyz'')` prüfen.

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

```
<papers>
  <p name="Parallel Computation of Generalized Hypertree Decompositions">
    <is_article>no</is_article>
    <is_conf>yes</is_conf>
  </p>
  <p name="Semantic Width Revisited">
    <is_article>no</is_article>
    <is_conf>yes</is_conf>
  </p>
  <p name="On the expressive power of collective attacks">
    <is_article>yes</is_article>
    <is_conf>no</is_conf>
  </p>
  <p name="Complexity of Abstract Argumentation under a Claim-Centric View.">
    <is_article>no</is_article>
    <is_conf>yes</is_conf>
  </p>
</papers>
```

Geben Sie nun die XQuery Abfrage an:

```
<papers>
  {
    for $p in doc("bibliography.xml")//title/text()
    where not(contains($p, "exam"))
    return <p name="{$p}">
    <is_article>{
      let $a := doc("bibliography.xml")//article/title[text() = $p]
      return if(count($a)>0) then "yes"
      else "no"
    }</is_article>
    <is_conf>{
      let $a := doc("bibliography.xml")//conference_paper/title[text() = $p]
      return if(count($a)>0) then "yes"
      else "no"
    }</is_conf>
    </p>
  }
</papers>
```

Aufgabe 7: (8)

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

Hinweis: Die `name()` *function liefert den Namen eines Elements.*

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

 <xsl:template match="bibliography">
  Registered Authors:
  <xsl:apply-templates select="//reg_author"/>
 </xsl:template>

 <xsl:template match="reg_author">
  <xsl:variable name="var" select="self::node()/@id" />
  Author: <xsl:value-of select="current()[first_name | name]"/>
  ==================================

  <xsl:for-each select ="//conference_paper[author/@ref = $var] |  //article[author/@ref = $var]  ">
    <xsl:call-template name="Citation" select="current()">
      <xsl:with-param name="title" select = "current()/title" />
      <xsl:with-param name="type" select = "name(.)" />
      <xsl:with-param name="type_name" select = "current()/conference | current()/journal" />
    </xsl:call-template>
  </xsl:for-each>

  <xsl:choose>
   <xsl:when test="not(//conference_paper[author/@ref = $var] | //article[author/@ref = $var])">
    No papers found.
   </xsl:when>
   <xsl:otherwise>
    Found <xsl:value-of select="count(//conference_paper[author/@ref = $var])"/>  conference paper(s).
    Found <xsl:value-of select="count(//article[author/@ref = $var])"/>  journal paper(s).
   </xsl:otherwise>
  </xsl:choose>
 </xsl:template>

 <xsl:template name="Citation">
  <xsl:param name = "title" /> <xsl:param name = "type"/> <xsl:param name = "type_name"/>
    Title: <xsl:value-of select="$title"/> .
    <xsl:value-of select="$type"/>: <xsl:value-of select="$type_name"/> .
    Authors:
    <xsl:choose>
     <xsl:when test="current()/author[1]/@ref">
    <xsl:value-of select="//reg_author[@id = current()/author[1]/@ref]" />
   </xsl:when>
   <xsl:otherwise>
    <xsl:value-of select="current()/author" />
   </xsl:otherwise>
  </xsl:choose>
  <xsl:if test="count(current()/author) > 2">
   et al.
  </xsl:if>
 </xsl:template>
</xsl:stylesheet>
```

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

Sie müssen sich nicht um Whitespaces kümmern.

```
Registered Authors:

    Author:   Cem  Okulmus
    ==================================

    Title: Parallel Computation of Generalized Hypertree Decompositions .
    conference_paper: AMW 2019 .
    Authors:  G. Gottlob et al.

    Title: SSD 2019-6 exam .
    article:  .
    Authors:   Wolfgang Dvorak et al.

            Found 1  conference paper(s).
            Found 1  journal paper(s).

    Author: Matthias  Lanzinger
    ==================================

    Title: Semantic Width Revisited .
    conference_paper: AMW 2019 .
    Authors: G. Gottlob et al.

    Title: SSD 2019-6 exam .
    article:  .
    Authors: Wolfgang Dvorak et al.

            Found 1  conference paper(s).
            Found 1  journal paper(s).

    Author: Wolfgang Dvorak
    ==================================

    Title: On the expressive power of collective attacks .
    article: Argument  & Computation .
    Authors: Wolfgang Dvorak et al.

    Title: SSD 2019-6 exam .
    article:  .
    Authors: Wolfgang Dvoraket al.

            Found 0  conference paper(s).
            Found 2  journal paper(s).
```

Total points: 75

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

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE bibliography SYSTEM "bibliography.dtd">
<bibliography>
  <conference_paper>
      The paper <title>Parallel Computation of Generalized Hypertree Decompositions</title> was published at
      <conference>AMW 2019</conference>
      <author>G. Gottlob</author>
      <author ref="a1">C. Okulmus</author>
      <author>R. Pichler</author>
  </conference_paper>
  <reg_author id="a1">
      <first_name>Cem</first_name>  <last_name>Okulmus</last_name>
  </reg_author>
  <conference_paper>
      <title>Semantic Width Revisited</title>
      <conference>AMW 2019</conference>
      <author>G. Gottlob</author>
      <author ref="a2"/>
      <author>R. Pichler</author>
  </conference_paper>
  <reg_author id="a2">
      <first_name>Matthias</first_name>  <last_name>Lanzinger</last_name>
  </reg_author>
  <reg_author id="a3">
      <name>Wolfgang Dvorak</name>
  </reg_author>
  <article>
      <title>On the expressive power of collective attacks</title>
      <journal>Argument  &amp;amp; Computation</journal>
      <author ref="a3">W. Dvorak</author>
      <author>J. Fandinno</author>
      <author>S. Woltran</author>
  </article>
  <conference_paper>
      <title>Complexity of Abstract Argumentation under a Claim-Centric View.</title>
      <conference>AAAI 2019</conference>
      <author>W. Dvorak</author>
      <author>J. Fandinno</author>
      <author>S. Woltran</author>
  </conference_paper>
   <article>
      <title>SSD 2019-6 exam</title>
      <type printed="yes">dbai exams</type>
      <author ref="a3">WD</author>
      <author ref="a2">ML</author>
      <author ref="a1">O</author>
  </article>
</bibliography>
```