PRÜFUNG IN "SEMI-STRUKTURIERTE DATEN" **184.705**			22. 10. 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="Q" type="QType">
         <xsd:key name="key1">
            <xsd:selector xpath=".//W/U"/>
            <xsd:field xpath="."/>
         </xsd:key>
    </xsd:element>
    <xsd:complexType name="QType">
        <xsd:sequence  maxOccurs="2">
            <xsd:element name="U" type="UType"/>
            <xsd:element name="W" type="WType"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:simpleType name="UType">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="([A-Z]+)|([A-Z]([a-z])+)"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType name="WType">
        <xsd:choice>
            <xsd:element name="W" type="WType"/>
            <xsd:element name="U" type="UType" maxOccurs="2" />
        </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. `<Q/>` valid ◯ invalid ⊗
2. `<Q><U>SSD</U><W><U>vu</U><U>exam</U></W></Q>` valid ◯ invalid ⊗
3. `<Q><U>SSD</U><W><U>DBS</U></W></Q>` valid ⊗ invalid ◯
4. `<Q><U>SSD</U><W><U>SSD</U></W></Q>` valid ⊗ invalid ◯
5. `<Q><U>DBS</U><W><U>SSD</U><U>SSD</U></W></Q>` valid ◯ invalid ⊗
6. `<Q><U>DBS</U><W><U>VO</U></W><U>DBS</U><W><U>UE</U></W></Q>` valid ⊗ invalid ◯
7. `<Q><U>SSD</U><U>DBS</U><W><U>VU</U></W><W><U>UE</U></W></Q>` valid ◯ invalid ⊗
8. `<Q><U>SSD</U><W><W><U>VU</U></W></W></Q>` 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. Wie verhält sich der Speicherbedarf von Tree-based Parsern wie DOM zur Größe des geparsten XML Dokuments?

   Antwort: Der Speicherbedarf steigt linear mit der Größe des geparsten XML Dokuments.

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

   Antwort: Event-based Parser, SAX.

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

   Antwort: Es wendet die Templates der Kinder an.

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

   Antwort: Es wird nur eines ausgeführt, dabei wir das spezifischeste Template ausgewählt.

5. Welches Datenmodell verwenden wir für semistrukturierte Daten?

   Antwort: Bäume mit Labels auf den Kanten.

6. Was versteht man unter FLWOR Expressions?

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

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

   Antwort: Wohl-geformte XML Dokumente entsprechen dem XML-Standard und valide Dokumente erfüllen zusätzlich noch die Bedingungen eines Schema das z.B. als DTD oder XML-Schema gegeben ist.

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

   Antwort: Die Werte der Schlüsselattribute müssen eindeutig über alle Schlüssel sein.

9. Auf welche Teile eines XML Dokuments wird der Default Namespace angewendet?

   Antwort: Auf Elemente ohne Prefix.

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

    Antwort: z.b.: Fix vorgegebene Menge von Elemente vs. frei definierbare Menge von Elemente

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

Aufgabe 3: (12)

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

- Das Wurzel Element hat drei Kind-Elemente `books`, `clients` und `lendings`, wobei jedes von diesen beliebig viele Kind-Elemente vom entsprechenden Typ (`book`, `client` oder `lending`) haben kann.
- Jedes `book` Element enthält ein `title` Element, mindestens ein `author` Element, ein `category` Element und möglicherweise ein `description` Element (in dieser Reihenfolge).
- Zusätzlich wird für Bücher immer ein Attribut mit einer eindeutigen Nummer gespeichert und ein weiteres Attribut mit der Seitenanzahl.
- `description` Elemente enthalten eine textuelle Beschreibung des Buches die Referenzen zu anderen Büchern in Form von `ref` Elementen enthalten kann. Diese `ref` Elemente enthalten ein Attribut `booknr` das auf die nr eines `book` Element verweist.
- Für `client` Elemente wird ein Attribut mit einer eindeutigen id gespeichert, und der Name als text content.
- Ausleihen werden in `lending` Elementen gespeichert wobei die ID des Kunden die Nummer des Buches und das Datum der Ausleihe gespeichert werden. Bei beendeten Ausleihen wird auch das Datum der Retournierung gespeichert.
- Wenn nicht angegeben treffen Sie plausible Annahmen über Typen von Attributen und Elementen.

File **library.dtd**:

```
<!ELEMENT library (books,clients,lendings)>

<!ELEMENT books (book)*>
<!ELEMENT clients (client)*>
<!ELEMENT lendings (lending)*>

<!ELEMENT book (title,author+,category,description?)>
<!ATTLIST book nr ID #REQUIRED pages CDATA #REQUIRED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT category (#PCDATA)>
<!ELEMENT description (#PCDATA|ref)*>
<!ELEMENT ref EMPTY>
 <!ATTLIST ref booknr IDREF #REQUIRED>
<!ELEMENT client (#PCDATA)>
<!ATTLIST client id ID #REQUIRED>

<!ELEMENT lending EMPTY>
<!ATTLIST lending clientid IDREF #REQUIRED
                  booknr IDREF #REQUIRED
                  date CDATA #IMPLIED
                  returned CDATA #IMPLIED>
```

Aufgabe 4: (10)

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

```
sum(//book[./description]/@pages)
```

```
700
```

```
//book[@nr=//lendings/lending/@booknr]/title/text()
```

```
Guards! Guards!
XML in a Nutshell
The Neverending Story
```

```
//lending[@clientid=//client[.='Bastian Bux']/@id][1]/@booknr
```

```
b5
```

```
//book[@nr=//lending/@booknr][3]/@nr
```

```
b6
```

```
//book[5][@nr=//lending/@booknr]/title
```

```
<title>XML in a Nutshell</title>
```

Aufgabe 5: (10)

Vervollständigen Sie die Methode `lending`, die drei Parameter `client`, `book` und `date` besitzt und mittels **DOM** folgende Änderungen am, in der Variable `doc` gespeicherten, Dokument durchführt:

Falls das Buch mit der `booknr`: `book` **bereits** von dem Kunden mit der `clientid`: `client` ausgeliehen ist, wird es mit dem Datum `date` retourniert. Dazu wird das `returned` Attribut auf `date` gesetzt.

Falls das Buch mit der `booknr`: `book` **nicht** von dem Kunden mit der `clientid`: `client` ausgeliehen ist, wird es mit dem Datum `date` ausgeliehen. Dazu wird ein neues `lending` Element erstellt und das `date` Attribut dieses neuen Elements wird auf `date` gesetzt. Vergessen Sie nicht, dass neue Element im bestehenden Dokument einzufügen.

```
private static XPath xPath = XPathFactory.newInstance().newXPath();
Document doc;

public void lending (String client, String book, String date) throws Exception {
    XPathExpression xpe = xPath.compile("//lending[@clientid = '" + client +
                                    "' and @booknr = '" + book + "' and not(@returned)]");

    NodeList list = (NodeList) xpe.evaluate(doc, XPathConstants.NODESET);

    if (list.getLength() == 1) {
        //find first to element
        Element lending = (Element) list.item(0);
        lending.setAttribute("returned", date);
    } else {
        xpe = xPath.compile("//lendings");

        NodeList lendings = (NodeList) xpe.evaluate(doc, XPathConstants.NODESET);
        Element lending = doc.createElement("lending");
        lending.setAttribute("clientid", client);
        lending.setAttribute("booknr", book);
        lending.setAttribute("date", date);

        lendings.item(0).appendChild(lending);
    }
}
```

Aufgabe 6: (8)

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

```
<author-info>
{
  for $a in distinct-values(doc("library.xml")//book/author)
  let $bs := doc("library.xml")//book[author/text()=$a]/@pages
  order by $a
  where sum($bs) > 600
  return <a>
  <name>{$a}</name>
  <pw>{sum($bs)}</pw>
  <rs>
  {
    for $l in doc("library.xml")//lending
    let $la := doc("library.xml")//book[@nr=$l/@booknr]
    let $c := doc("library.xml")//client[@id=$l/@clientid]
    order by $c/text()
    where $la/author=$a
    return <r>{$c/text()}</r>
  }
  </rs>
  </a>
}
</author-info>
```

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

Sie müssen sich nicht um Whitespaces kümmern.

```
<author-info>
    <a>
        <name>Eickler, André</name>
        <pw>880</pw>
        <rs/>
    </a>
    <a>
        <name>Elliotte Rusty Harold</name>
        <pw>690</pw>
        <rs>
            <r>Arthur Dent</r>
            <r>Bastian Bux</r>
            <r>Phryne Fisher</r>
            <r>Sophie Haas</r>
        </rs>
    </a>
    <a>
        <name>Kemper, Alfons</name>
        <pw>880</pw>
        <rs/>
    </a>
    <a>
        <name>Terry Pratchett</name>
        <pw>700</pw>
        <rs>
            <r>Arthur Dent</r>
        </rs>
    </a>
    <a>
        <name>W. Scott Means</name>
        <pw>690</pw>
        <rs>
            <r>Arthur Dent</r>
            <r>Bastian Bux</r>
            <r>Phryne Fisher</r>
            <r>Sophie Haas</r>
        </rs>
    </a>
</author-info>
```

Aufgabe 7: (8)

Betrachten Sie folgendes XSLT Dokument **library.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="library">
        <xsl:apply-templates select="//client[@id = 'p3' or @id = 'p4']" />
    </xsl:template>

    <xsl:template match="client">
        <xsl:variable name="lender" select="@id" />
        ---
        <xsl:value-of select="."/>'s transactions:

        <xsl:for-each select="../..//lending[@clientid=$lender]">
        Borrowed on <xsl:value-of select="current()/@date"/>
        the book <xsl:apply-templates select="../..//book[@nr=current()/@booknr]"/>
        <xsl:apply-templates select="."/>
        </xsl:for-each>
        ---
    </xsl:template>

    <xsl:template match="book">
        "<xsl:value-of select="title"/>"
        written by <xsl:if test="author/following-sibling::author">the authors: </xsl:if>
        <xsl:for-each select="author">
        <xsl:value-of select="."/>
        <xsl:if test="following-sibling::author"> and </xsl:if>
        </xsl:for-each>
    </xsl:template>

    <xsl:template  match="lending[@returned]">
        and has returned the book on <xsl:value-of select="./@returned"/> .
    </xsl:template>

    <xsl:template  match="lending[not(@returned)]">
        and has not yet returned the book.
    </xsl:template>
</xsl:stylesheet>
```

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

Sie müssen sich nicht um Whitespaces kümmern.

```
---
Arthur Dent’s transactions:


Borrowed on 20.5.2018
the book
"XML in a Nutshell"
written by the authors:
Elliotte Rusty Harold and W. Scott Means
and has returned the book on 22.6.2018 .

Borrowed on 2.3.2018
the book
"Guards! Guards!"
written by
Terry Pratchett
and has not yet returned the book.

---

---
Phryne Fisher’s transactions:


Borrowed on 25.6.2018
the book
"XML in a Nutshell"
written by the authors:
Elliotte Rusty Harold and W. Scott Means
and has not yet returned the book.

---
```

Total points: 75

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

```
<library>
    <books>
        <book nr="b1" pages="320">
            <title>Guards! Guards!</title>
            <author>Terry Pratchett</author>
            <category>Fantasy</category>
            <description>Fantasy novel in the Discworld series</description>
        </book>
        <book nr="b2" pages="380">
            <title>Men at Arms</title>
            <author>Terry Pratchett</author>
            <category>Fantasy</category>
            <description> Follow up to <ref booknr="b1"/> </description>
        </book>
        <book nr="b3" pages="450">
            <title>XQuery</title>
            <author>Priscilla Walmsley</author>
            <category>Science</category>
        </book>
        <book nr="b4" pages="880">
            <title>Datenbanksysteme</title>
            <author>Kemper, Alfons</author>
            <author>Eickler, André</author>
            <category>Science</category>
        </book>
        <book nr="b5" pages="690">
            <title>XML in a Nutshell</title>
            <author>Elliotte Rusty Harold</author>
            <author>W. Scott Means</author>
            <category>Science</category>
        </book>
        <book nr="b6" pages="530">
            <title>The Neverending Story</title>
            <author>Michael Ende</author>
            <category>Fantasy</category>
        </book>
    </books>
    <clients>
        <client id="p1">Bastian Bux</client>
        <client id="p2">Sophie Haas</client>
        <client id="p3">Arthur Dent</client>
        <client id="p4">Phryne Fisher</client>
    </clients>
    <lendings>
        <lending clientid="p2" booknr="b5" date="1.3.2018" returned="15.4.2018"/>
        <lending clientid="p1" booknr="b5" date="16.4.2018" returned="14.5.2018"/>
        <lending clientid="p1" booknr="b6" date="20.5.2018"/>
        <lending clientid="p3" booknr="b5" date="20.5.2018" returned="22.6.2018"/>
        <lending clientid="p4" booknr="b5" date="25.6.2018"/>
        <lending clientid="p3" booknr="b1" date="2.3.2018"/>
    </lendings>
</library>
```