PRÜFUNG IN "SEMI-STRUKTURIERTE DATEN" **184.705**			02.12.2019
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="A" type="typeA"/>

  <xsd:complexType name="typeA">
    <xsd:choice>
      <xsd:sequence>
        <xsd:element name="N" type="nr"/>
        <xsd:element name="A" minOccurs="0" maxOccurs="2" type="typeA"/>
      </xsd:sequence>
      <xsd:sequence maxOccurs="2">
        <xsd:element name="A" type="typeA"/>
        <xsd:element name="N" type="xsd:int"/>
      </xsd:sequence>
    </xsd:choice>
  </xsd:complexType>

  <xsd:simpleType name="nr">
      <xsd:restriction base="xsd:string">
         <xsd:pattern value="[e,a,h][0-9]+"/>
      </xsd:restriction>
  </xsd:simpleType>
</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. `<A/>` valid ○ invalid ○
2. `<A><N>e034h223</N></A>` valid ○ invalid ○
3. `<A><N>e00325673</N></A>` valid ○ invalid ○
4. `<A> <A><N>e0</N></A> <N>1024</N> <A><N>e0</N></A> <N>42</N> </A>` valid ○ invalid ○
5. `<A> <N>e0</N> <A><N>e0</N></A> <N>42</N> </A>` valid ○ invalid ○
6. `<A> <N>e0</N> <A><N>e0</N></A> <A><N>h42</N></A> </A>` valid ○ invalid ○

(Für jede korrekte Antwort 2 Punkte, **für jede falsche Antwort -2 Punkte**, unbeanwortete Fragen 0 Punkte, 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. Wie Unterscheiden sich Elemente von Attributen bezüglich der Signifikanz der Reihenfolge im XML Dokument?

3. Welches Datenmodell verwenden wir für semistrukturierte Daten?

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

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

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

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

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

Aufgabe 3: (12)

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

- Das Wurzel Element enthält ein `meta-data` und ein `content` Element.
- Das `meta-data` Element enthält genau ein `title` Element, mindestens ein `author` Element und beliebig viele `keyword` Elemente (in dieser Reihenfolge).
- `author` Elemente bestehen entweder a) aus einem `name` Element oder (b) aus einem oder mehreren `first` Element gefolgt von einem `last` Element. Jedes `author` Element hat ein Attribut mit einer eindeutigen ID.
- Das `content` Element enthält optional ein `preface` Element, mindestens ein `chapter` Element und beliebig viele `appendix` Elemente. `preface`, `chapter` und `appendix` Elemente enthalten Text und verweisen mit `authorref` auf Autoren und mit `ref` auf andere Teile des Buches.

  `chapter` und `appendix` Elemente haben immer ein Attribut `title`. `appendix` Elemente haben immer eine eindeutige `id` bei `chapter` Elementen ist dieses optional.
- Wenn nicht angegeben treffen Sie plausible Annahmen über Typen von Attributen und Elementen.

File **handbook.dtd**:

Aufgabe 4: (10)

Betrachten Sie die folgenden XPath-Abfragen angewandt auf das Dokument **handbook.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(//.[@id="a2"])
```

```
//content/*[4]
```

```
//author[@id=//content//authorref/@id][3]/*[1]
```

```
//chapter[following-sibling::chapter[@id]]/@title
```

```
//chapter[@id=following::*/ref/@id]/@id
```

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 **handbook.xml** enthält.

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

    for(int i=0; i < nodes_author.getLength(); i++) {
        Node tnode = nodes_author.item(i);

        Node last_child = tnode.getLastChild();
        String[] nameparts = last_child.getTextContent().split("␣");
        String name = nameparts[nameparts.length-1];

        NodeList nodes_contentList = doc.getElementsByTagName("content");
        Node content = nodes_contentList.item(0);
        Node hbPart= content.getFirstChild();
        String firstMention = null;
        for(;hbPart != null; hbPart = hbPart.getNextSibling()) {
            Node current = hbPart;
            if (current != null && current.getTextContent().contains(name)) {
                if (firstMention == null) {
                    firstMention = current.getTextContent();
                }
            }
        }
        System.out.println (name + ",␣" + firstMention );
    }
}
```

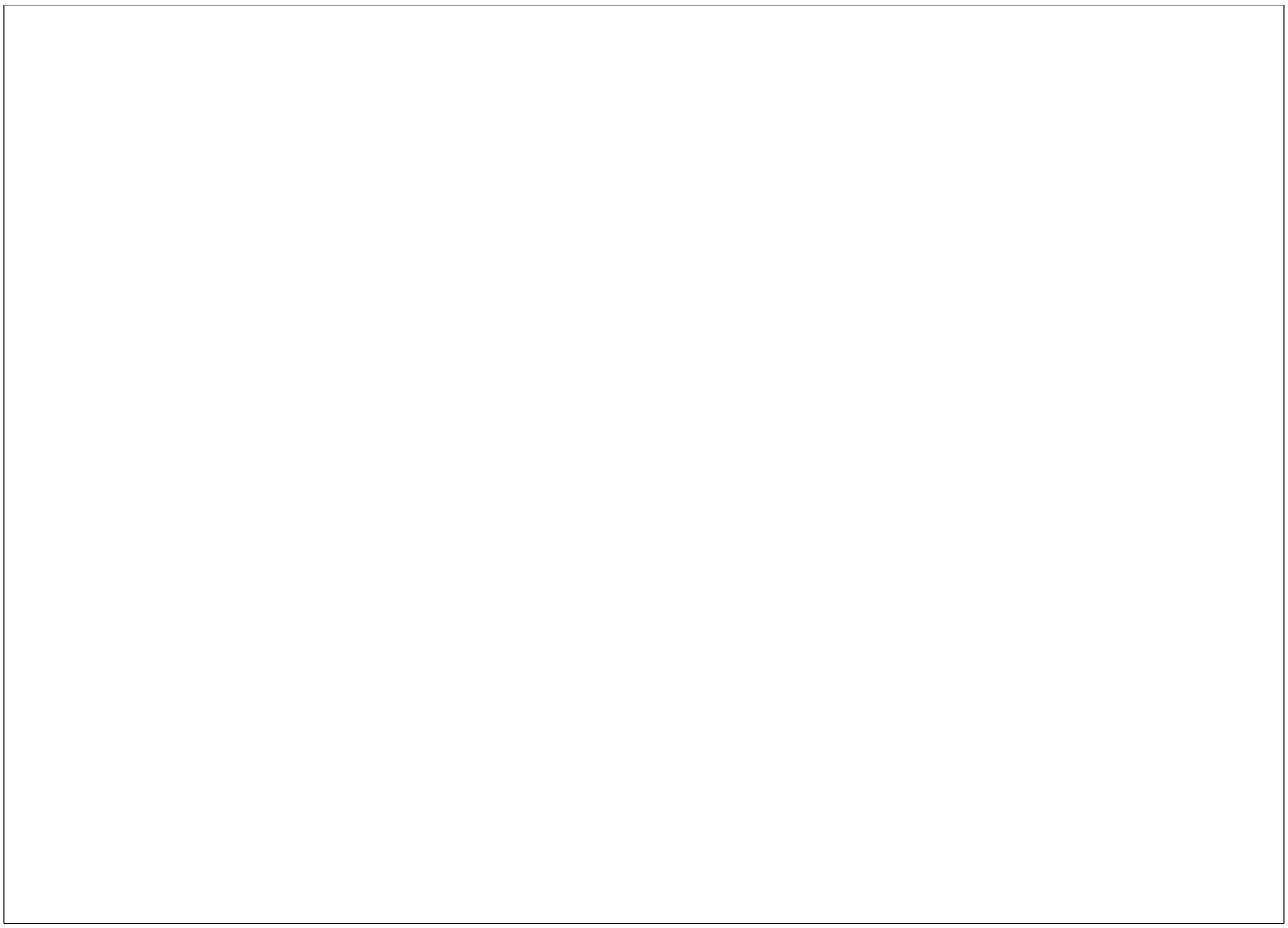
Aufgabe 6: (8)

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

```
<info>
{
  for $p in doc("handbook.xml")//content/*
  where $p/name() != "chapter"
  order by $p/name()
  return <pa title="{$p/@title}">
  {
    let $r := distinct-values($p/authorref/@id)
    let $r2 := distinct-values($p/ref/@id)
    return <nr>{count($r)+count($r2)}</nr>
  }
  {
    if(count($p/@id) = 0) then
      <warn>id warning</warn>
    else
      <ok/>
  }</pa>
}</info>
```

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

Sie müssen sich nicht um Whitespaces kümmern.



Aufgabe 7: (10)

Betrachten Sie folgende Text Ausgabe **handbook.xslt.out**:

```
Summary

Preface
mentions "A.Dent", referring to Arthur Dent
mentions "Scotty", referring to Montgomery Scott

Chapter part 1
mentions "Kaylee Frye", referring to Kaywinnet, Lee Frye
cites Appendix A

Chapter part 2
cites Appendix B
```

Es handelt sich um eine Zusammenfassung des Handbuches, wo für das Preface, und jedes Kapitel gelistet wird welche Personen genannt werden, und auf welche Appendizes verwiesen wird. Bei den Personen sollen sowohl der Name im Text, als auch der vollständige Name erwähnt werden.

Wichtig: Es sollen nur jene Kapitel in dieser Zusammenfassung aufscheinen, die auch irgendwelche Verweise oder Referenzen haben, Kapitel ohne solche `ref` oder `authorref` Elemente sollen übersprungen werden.

Geben Sie nun ein XSLT Dokument an, das angewandt auf **handbook.xml**, die Ausgabe **handbook.xslt.out** produziert. Es geht hierbei darum, das in der Angabe beschriebene Verhalten im Bezug auf den Inhalt von **handbook.xml** zu reproduzieren, z.B. sollte ihr XSLT Dokument **nicht** statisch immer dieselbe Ausgabe produzieren, auch wenn die XML Datei grundlegend verändert werden würde.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="text"/>









</xsl:stylesheet>
```

Total points: 75

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

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE handbook SYSTEM "handbook.dtd">
<handbook>
    <meta-data>
        <title>The handbook of Science fiction</title>
        <author id="a1">
            <name>Arthur Dent</name>
        </author>
        <author id="a2">
            <first>Montgomery</first> <last>Scott</last>
        </author>
        <author id="a3">
            <first>Kaywinnet</first> <first>Lee</first> <last>Frye</last>
        </author>
        <keyword>science</keyword><keyword>fiction</keyword>
    </meta-data>
    <content>
        <preface>
        This textbook is co-authored by <authorref id="a1">A. Dent</authorref>
        and <authorref id="a2">Scotty</authorref> and is organized in four chapters.
        </preface>
        <chapter title="part 1">
            This chapter is based on research by <authorref id="a3">Kaylee Frye</authorref>.
            The corresponding experimental data is provided in <ref id="app1"/>.
        </chapter>
        <chapter title="part 2"  id="c2">
            The mathematical background for this chapter is discussed in
            <ref id="app2"/>
        </chapter>
        <chapter title="part 3"/>
        <chapter title="part 4"/>
        <appendix id="app1" title="A">
        some data
        </appendix>
        <appendix id="app2" title="B">
            mathematical foundations of chapter <ref id="c2"/>.
        </appendix>
    </content>
</handbook>
```