

Gruppe A	PRÜFUNG AUS "SEMISTRUKTURIERTE DATEN" 184.705		25. 10. 2013
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 die folgende DTD Datei **test.dtd**:

```
<!ELEMENT A ((A, B) | (C, A)*)>
<!ELEMENT B (#PCDATA | C)*>
<!ELEMENT C EMPTY>
<!ATTLIST A key ID #REQUIRED>
<!ATTLIST C ref IDREF #IMPLIED>
```

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.dtd** zu entscheiden.

Kreuzen Sie an, welche der folgenden XML-Dateien gültig bezüglich **test.dtd** sind.

- | | | |
|---|---|---|
| 1. <A/> | gültig <input type="radio"/> | ungültig <input checked="" type="radio"/> |
| 2. <C/> | gültig <input type="radio"/> | ungültig <input checked="" type="radio"/> |
| 3. <C/> | gültig <input checked="" type="radio"/> | ungültig <input type="radio"/> |
| 4. <C/>text | gültig <input type="radio"/> | ungültig <input checked="" type="radio"/> |
| 5. text | gültig <input type="radio"/> | ungültig <input checked="" type="radio"/> |
| 6. text | gültig <input checked="" type="radio"/> | ungültig <input type="radio"/> |
| 7. <C ref="x01"/> | gültig <input checked="" type="radio"/> | ungültig <input type="radio"/> |
| 8. <C ref="x02"/> | gültig <input checked="" type="radio"/> | ungültig <input type="radio"/> |

(Pro korrekter Antwort 1.5 Punkte, **pro inkorrekter Antwort -1.5 Punkte**, pro nicht beantworteter Frage 0 Punkte, für die gesamte Aufgabe mindestens 0 Punkte)

Aufgabe 2:

(15)

Kreuzen Sie an, ob die folgenden Aussagen wahr oder falsch sind.

1. XML gibt es schon wesentlich länger als das WWW. wahr falsch
2. Wohlgeformte XML-Dokumente sind immer auch gültig. wahr falsch
3. Eine XML Schema Definition spezifiziert das Wurzelement des XML-Instanz Dokuments. wahr falsch
4. Pro XML-Schema Datei ist maximal ein Target-Namespace erlaubt. wahr falsch
5. DOM ist eine W3C-Recommendation. wahr falsch
6. SAX Filter dürfen verschachtelt verwendet werden. wahr falsch
7. Der Speicherbedarf eines SAX-Parsers ist im Allgemeinen geringer als eines DOM-Parsers wahr falsch
8. Bei XSLT wird pro Knoten maximal ein Template angewendet. wahr falsch
9. Das Resultat eines XPATH Ausdrucks ist immer in Dokument-Order. wahr falsch
10. XQUERY baut auf einem relationalem Datenmodell auf. wahr falsch

(Pro korrekter Antwort 1.5 Punkte, **pro inkorrektter Antwort -1.5 Punkte**, pro nicht beantworteter Frage 0 Punkte, für die gesamte Aufgabe mindestens 0 Punkte)

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

Aufgabe 3:

(12)

Vervollständigen Sie die XML Schema Datei `rechnungen.xsd`, sodass XML-Dokumente in der Gestalt von `rechnungen.xml` (siehe Anhang) bezüglich dieses Schemas gültig sind. Berücksichtigen Sie beim Erstellen des Schemas folgende Punkte:

- Das Element `rechnungen` ist das Wurzelement und besteht aus mindestens einem `rechnung`-Element.
- Das Element `rechnung` besitzt zwei verpflichtende Attribute (`inhalt` und `datum`). Der Elementinhalt besteht aus Text und den Subelementen `betrag`, `spesen` und `total`. Diese sollen folgendermaßen auftreten:
- Es sollen beliebig viele `betrag`-Elemente erlaubt sein; jedem `betrag`-Element kann optional ein `spesen`-Element nachfolgen. Abschließend soll ein `total`-Element auftreffen, jedoch nur dann wenn zumindest ein `betrag`-Element (innerhalb des `rechnung`-Elements) vorhanden ist
- Die Elementinhalte von `betrag`, `spesen` und `total` sollen nicht-negative Zahlen mit 2 Nachkommastellen sein. Versuchen Sie diesen Typ selbst zu spezifizieren. Hinweis: Leiten Sie Ihren Typ von `xsd:decimal` ab.
- Es sind keine Schlüssel zu definieren.

Datei `rechnungen.xsd`:

```
<!-- Sie haben auch noch auf der folgenden Seite Platz! -->

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:element name="rechnungen">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="rechnung" maxOccurs="unbounded">
          <xsd:complexType mixed="true">
            <xsd:sequence minOccurs="0">
              <xsd:sequence maxOccurs="unbounded">
                <xsd:element name="betrag" type="wert"/>
                <xsd:element name="spesen" minOccurs="0" type="wert"/>
              </xsd:sequence>
              <xsd:element name="total" type="wert"/>
            </xsd:sequence>
            <xsd:attribute name="datum" type="xsd:date" use="required"/>
            <xsd:attribute name="kundennummer" type="xsd:integer" use="required"/>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:simpleType name="wert">
    <xsd:restriction base="xsd:decimal">
      <xsd:minExclusive value="0"/>
      <xsd:fractionDigits value="2"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```

Datei **rechnungen.xsd** (Fortsetzung):

Aufgabe 4:

(10)

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

- Falls als Ergebnis mehrere Knoten selektiert werden, trennen Sie die jeweiligen Ausgaben durch Leerzeichen.
- Falls der angegebene XPath Ausdruck keine Knoten selektiert, notieren Sie im entsprechenden Feld "leere Ausgabe".

Betrachten Sie dazu folgendes Beispiel:

```
//rechnung/@datum
```

```
2013-10-10    2013-10-11    2013-10-12
```

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

```
//rechnung/@kundennummer
```

```
111    4711    111
```

```
//rechnung[@kundennummer='111']/@datum
```

```
2013-10-10    2013-10-12
```

```
//rechnung[not(total)]/@kundennummer
```

```
111
```

```
count(//betrag)
```

```
3
```

```
//total/preceding::*/@datum
```

```
2013-10-10
```

Aufgabe 5:

(8)

Betrachten Sie folgende-XQuery Abfrage **rechnungen.xq** angewandt auf **rechnungen.xml**:

```
for $r in //rechnung
let $s := sum($r/spesen)
let $b := sum($r/betrag)
where $s gt 0
order by $s
return <r>{$b}, {$s}</r>
```

Geben Sie nun die Ausgabe von **rechnungen.xq** angewandt auf **rechnungen.xml** an.
Die exakte Behandlung von Whitespaces ist dabei nicht relevant.

```
<r>38, 0.5</r>
<r>50, 5</r>
```

Aufgabe 6:

(9)

Erstellen Sie ein XSLT-Stylesheet **rechnungen.xsl**, das angewandt auf Dokumente der Gestalt **rechnungen.xml** folgende Ausgabe erzeugt:

- Es soll nur die **erste** Rechnung verarbeitet werden.
- Für diese Rechnung soll das Element **html** erzeugt werden. Innerhalb eines **title** Elements soll das Rechnungsdatum ausgegeben werden. Der Inhalt der Rechnung soll weitestgehend unverändert ausgegeben werden.
- Einzige Änderung ist, dass für **betrag**, **spesen** und **total** Elemente in der Ausgabe **b** Elemente erzeugt werden. Deren Inhalt soll unverändert ausgegeben werden.

Hinweis: Die weitestgehend unveränderte Ausgabe wird am einfachsten durch die Built-In Templates erzielt. Sie müssen also nur dafür sorgen, dass diese auch aufgerufen werden.

Für das Dokument **rechnungen.xml** soll beispielsweise folgende Ausgabe erzeugt werden:

```
<html>
  <title>2013-10-10</title>

  Liebes SSD-Team,

  fuer die 50 Kopien der Pruefungsboegen verrechnen
  wir ihnen <b>50.00</b> Euro zuzueglich
  <b>5.00</b> Spesen fuer das Expresservice.

  Der offene Betrag von <b>55.00</b> ist unverzueglich
  zu zahlen.

  MfG, Kopierzentrum copy/paste.

</html>
```

Vervollständigen Sie hier das XSLT-Stylesheet **rechnungen.xsl**. Die Verwendung von Kontrollstrukturen wie `xsl:for-each`, `xsl:if`, etc. ist für die Lösung *nicht* erlaubt (und auch nicht sinnvoll)! Sie brauchen sich nicht um Whitespaces etc. zu kümmern.

Datei **rechnungen.xsl**:

```
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>

  <xsl:template match="/">
    <xsl:apply-templates select="rechnungen/rechnung[1]"/>
  </xsl:template>

  <xsl:template match="rechnung">
    <html>
      <title><xsl:value-of select="@datum"/></title>
      <xsl:apply-templates/>
    </html>
  </xsl:template>

  <xsl:template match="betrag|spesen|total">
    <b>
      <xsl:apply-templates/>
    </b>
  </xsl:template>
</xsl:stylesheet>
```

Vervollständigen Sie den folgenden SAX Handler, der angewandt auf Dokumente der Gestalt **rechnungen.xml** für jede Rechnung prüfen soll, ob der Totalbetrag (Element **total**) tatsächlich der Summe der Beträge und Spesen entspricht. Falls die Werte voneinander abweichen, soll auf den Standardoutput die Meldung "Total falsch berechnet!" ausgegeben werden.

Für das Dokument **rechnungen.xml** wird beispielsweise keine Meldung ausgegeben ($50+5 = 55$ bzw. $33+5+0.50 = 38.50$).

Denken Sie daran, dass **character** Events nicht immer den gesamten Textinhalt auf einmal zurückliefern!

```
public class CheckTotal extends DefaultHandler {

    float ist = 0;
    boolean in = false;
    StringBuffer sb = new StringBuffer();

    public void startElement(String uri, String localName, String qName, Attributes atts)
        throws SAXException {
        if ("betrag".equals(localName) || "spesen".equals(localName)
            || "total".equals(localName)) {
            in = true;
        }
    }

    public void endElement(String uri, String localName, String qName) throws SAXException {
        if ("betrag".equals(localName) || "spesen".equals(localName)) {
            in = false;
            ist += Float.parseFloat(sb.toString());
            sb.delete(0, sb.length());
        }
        if ("total".equals(localName)) {
            in = false;
            int soll = Float.parseFloat(sb.toString());
            if (ist != soll) {
                System.out.println("Total falsch berechnet!");
            }
            ist = 0;
        }
    }

    public void characters(char[] ch, int start, int length) throws SAXException {
        if (inPos) {
            sb.append(ch, start, length);
        }
    }
}
```


Sie können diese Seite abtrennen!

Datei rechnungen.xml:

```
<rechnungen>
  <rechnung datum="2013-10-10" kundenummer="111">
    Liebes SSD-Team,

    fuer die 50 Kopien der Pruefungsboegen verrechnen
    wir ihnen <betrag>50.00</betrag> Euro zuzueglich
    <spesen>5.00</spesen> Spesen fuer das Expressservice.

    Der offene Betrag von <total>55.00</total> ist unverzueglich
    zu zahlen.

    MfG, Kopierzentrum copy/paste.
  </rechnung>
  <rechnung datum="2013-10-11" kundenummer="4711">
    Lieber Herr Diplomand,

    fuer die 3 Kopien Ihrer Diplomabert verrechnen
    wir ihnen <betrag>33.00</betrag> Euro spesenfrei.

    Die Kopie Ihrer Zeugnisse macht <betrag>5.00</betrag>
    aus zzgl. Dokumentspesen von <spesen>0.50</spesen>.

    Bitte um baldige Begleichung des Gesamtbetrags von
    <total>38.50</total>.

    Alles Gute zu Ihrer baldigen Sponson.

    Ihr
    Kopierzentrum copy/paste.
  </rechnung>
  <rechnung datum="2013-10-12" kundenummer="111">
    Liebes SSD-Team,

    Zahlen Sie bitte endlich!

    Kopierzentrum copy/paste.
  </rechnung>
</rechnungen>
```

Gesamtpunkte: 75