

Gruppe A	PRÜFUNG AUS "SEMISTRUKTURIERTE DATEN" 184.705		1. 12. 2014
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?,B)>
<!ELEMENT B (#PCDATA | A | C)*>
<!ELEMENT C EMPTY>
<!ATTLIST A key ID #REQUIRED>
<!ATTLIST C choice (a|b|c|d) #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 (nehmen Sie an, dass **A** als Wurzelement spezifiziert ist):

1. <A/> gültig ungültig
2. <C/>xyz gültig ungültig
3. gültig ungültig
4. <C choice="e"/> gültig ungültig
5. <C choice="a"/><C choice="b"/><C choice="c"/> gültig ungültig
6. <C/> gültig ungültig
7. gültig ungültig
8. gültig ungültig

(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)

Aufgabe 2:

(15)

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

1. DTD steht für Data Type Definition. wahr falsch
2. Für jedes XML Dokument gilt: pro Element sind maximal 2 Kind-Elemente erlaubt. wahr falsch
3. XML Schemas erlauben typischerweise eine exaktere Spezifikation als DTDs. wahr falsch
4. In einem XML-Schema kann `xsd:attribute` ein Kind-Element von `xsd:simpleType` sein. wahr falsch
5. Der XPath Ausdruck “`./../* eq .`” liefert in jedem Knoten (außer der Wurzel) *wahr*. wahr falsch
6. Das DOM ist eine W3C Recommendation. wahr falsch
7. Bei einem Push Parser wird das XML-Dokument mehrmals durchlaufen. wahr falsch
8. Bei XSLT werden (mittels XPath) Knoten ausgewählt, auf die ein Template angewendet wird. wahr falsch
9. XSLT ist hauptsächlich eine imperative Programmiersprache. wahr falsch
10. XQuery erlaubt keine rekursiven Funktionsdeklarationen. 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 `schach.xml`, das Sie auf der letzten Seite dieser Prüfungsangabe finden.

Aufgabe 3:

(12)

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

- Das Element `schach` beinhaltet mindestens ein `eröffnung` Element. Jedes `eröffnung` Element hat ein Attribut `name` und genau ein Kindelement `weiß`.
- Elemente `weiß` haben gemischten Inhalt und potentiell mehrere Kindelemente `schwarz`. Elemente `schwarz` sollen keinen gemischten Inhalt haben und maximal ein Kindelement `weiß` besitzen.
- Definieren Sie die Attribute der Elemente `weiß` und `schwarz` entsprechend. Hinweise: Nutzen Sie `attributeGroup` um Schreibaufwand zu sparen.
- Sollten bei bestimmten Elementen oder Attributen keine näheren Angaben bezüglich des genauen Typs vorgegeben sein, wählen Sie selbst einen sinnvollen Typ aus.

Datei `schach.xsd`:

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

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

```
  <xs:element name="schach">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="eröffnung" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="weiß" type="weiß_typ"/>
            </xs:sequence>
            <xs:attribute name="name" type="xs:string" use="required"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="weiß_typ" mixed="true">
    <xs:sequence>
      <xs:element name="schwarz" type="schwarz_typ" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attributeGroup ref="zug"/>
  </xs:complexType>

  <xs:complexType name="schwarz_typ">
    <xs:sequence>
      <xs:element name="weiß" type="weiß_typ" minOccurs="0"/>
    </xs:sequence>
    <xs:attributeGroup ref="zug"/>
  </xs:complexType>
```

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

```
<xs:attributeGroup name="zug">
  <xs:attribute name="von" type="xs:string" use="required"/>
  <xs:attribute name="nach" type="xs:string" use="required"/>
  <xs:attribute name="schlägt" type="xs:boolean"/>
</xs:attributeGroup>
</xs:schema>
```

Aufgabe 4:

(10)

Betrachten Sie die folgenden XPath-Abfragen angewandt auf das Dokument **schach.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".
- Falls als Ergebnis **weiß** oder **schwarz** Knoten selektiert werden, geben Sie deren **nach** Attribute an.

Betrachten Sie dazu folgendes Beispiel:

```
//weiß
```

```
c4 d4 c4 e4 f3 b5
```

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

```
//schwarz[@von='c7']
```

```
c5
```

```
//schwarz[@nach='f6']/preceding-sibling::*
```

```
c5 e5
```

```
//schwarz[@nach='d5']//*
```

```
c4 c4 e6
```

```
//schwarz[@nach='e6']/ancestor::*
```

```
d4 d5 c4
```

```
//schwarz[ancestor::weiß[@nach='e4']]
```

```
e5 c6
```

Aufgabe 5:

(8)

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

```
let $zug := (//schwarz | //weiß)
for $feld in distinct-values($zug/@nach/string())
where starts-with($feld, 'c') or starts-with($feld, 'e')
order by $feld
return <feld name="{ $feld }" count="{count($zug[@nach=$feld])}"/>
```

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

```
<feld name="c4" count="3"/>
<feld name="c5" count="1"/>
<feld name="c6" count="1"/>
<feld name="e4" count="1"/>
<feld name="e5" count="2"/>
<feld name="e6" count="1"/>
```

Aufgabe 6:

(10)

Erstellen Sie ein XSLT-Stylesheet **schach.xsl**, das angewandt auf Dokumente der Gestalt **schach.xml** die Zugfolge ausgibt.
Für das Dokument **schach.xml** soll beispielsweise folgende Ausgabe erzeugt werden:

```
== Englische Eröffnung ==
c2-c4 | c7-c5
```

```
== Damengambit ==
d2-d4 | d7-d5
c2-c4 | d5-c4
```

```
== Spanische Eröffnung ==
e2-e4 | e7-e5
g1-f3 | b8-c6
f1-b5 |
```

Das bedeutet also:

- Für jede **eröffnung** soll der Name ausgegeben werden (davor und danach durch == abgegrenzt)
- Für jeden Zug von **weiß** soll das **von** Attribut und **nach** Attribut (getrennt durch -) ausgegeben werden. Nach jedem Zug von Weiß soll das Zeichen | ausgegeben werden, und die Verarbeitung bei dem **ersten** Schwarz-Kindelement fortgesetzt werden
- Für jeden Zug von **schwarz** soll das **von** Attribut und **nach** Attribut (getrennt durch -) ausgegeben werden. Nach jedem Zug von Schwarz soll ein Zeilenumbruch (
) ausgegeben werden

Vervollständigen Sie hier das XSLT-Stylesheet **schach.xml**.

Datei **schach.xml**:

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

  <xsl:template match="eröffnung">
    <xsl:text>&#xA;&#xA;== </xsl:text>
    <xsl:value-of select="@name"/>
    <xsl:text> ==&#xA;</xsl:text>
    <xsl:apply-templates/>
  </xsl:template>

  <xsl:template match="weiß">
    <xsl:value-of select="@von"/>
    <xsl:text>-</xsl:text>
    <xsl:value-of select="@nach"/>
    <xsl:text> | </xsl:text>
    <xsl:apply-templates select="schwarz[1]"/>
  </xsl:template>

  <xsl:template match="schwarz">
    <xsl:value-of select="@von"/>
    <xsl:text>-</xsl:text>
    <xsl:value-of select="@nach"/>
    <xsl:text>&#xA;</xsl:text>
    <xsl:apply-templates/>
  </xsl:template>

  <xsl:template match="text()"/>
</xsl:stylesheet>
```

Aufgabe 7:

(8)

Vervollständigen Sie den folgenden SAX Handler, der angewandt auf Dokumente der Gestalt **schach.xml** für jede **eröffnung** die Anzahl der Züge von **weiß** zählen soll (d.h. die Anzahl der **weiß** Nachfahren).

Für das Dokument **schach.xml** wird beispielsweise folgende Meldung (auf die Standardausgabe) ausgegeben:

Englische Eröffnung: 1

Damengambit: 2

Spanische Eröffnung: 3

Um die genaue Formatierung der Ausgabe brauchen Sie sich nicht zu kümmern

```
public class CountTotals extends DefaultHandler {

    int count;
    String current;

    public void startElement(String uri, String localName, String qName, Attributes atts)
        throws SAXException {
        if ("eröffnung".equals(localName)) {
            current = atts.getValue("name");
            count = 0;
        }
        if ("weiß".equals(localName)) {
            count++;
        }
    }

    public void endElement(String uri, String localName, String qName) throws SAXException {
        if ("eröffnung".equals(localName)) {
            System.out.println(current, count);
        }
    }
}
```

Sie können diese Seite abtrennen!

Datei schach.xml:

```
<schach>
  <eröffnung name="Englische Eröffnung">
    <weiß von="c2" nach="c4">
      Englische Symmetrievariante:
      <schwarz von="c7" nach="c5"/>
      Sizilianisch im Anzuge:
      <schwarz von="e7" nach="e5"/>
      Neutraler Entwicklungszug:
      <schwarz von="g8" nach="f6"/>
    </weiß>
  </eröffnung>

  <eröffnung name="Damengambit">
    <weiß von="d2" nach="d4">
      <schwarz von="d7" nach="d5">
        <weiß von="c2" nach="c4">
          Angenommenes Damengambit:
          <schwarz von="d5" nach="c4" schlägt="true"/>
          Abgelehntes Damengambit:
          <schwarz von="e7" nach="e6"/>
        </weiß>
      </schwarz>
    </weiß>
  </eröffnung>

  <eröffnung name="Spanische Eröffnung">
    <weiß von="e2" nach="e4">
      <schwarz von="e7" nach="e5">
        <weiß von="g1" nach="f3">
          <schwarz von="b8" nach="c6">
            <weiß von="f1" nach="b5"/>
          </schwarz>
        </weiß>
      </schwarz>
    </weiß>
  </eröffnung>
</schach>
```

Gesamtpunkte: 75