

Gruppe A	PRÜFUNG AUS "SEMISTRUKTURIERTE DATEN" 181.135		28. 10. 2011
Kennnr.	Matrikelnr.	Familienname	Vorname

Arbeitszeit: 120 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:

(9)

Betrachten Sie die folgende DTD Datei **test.dtd**:

```
<!ELEMENT A (B, C?)>
<!ELEMENT B (#PCDATA | C)*>
<!ELEMENT C (A | D)*>
<!ELEMENT D (A* | B* | C* | D*)>
<!ATTLIST D att1 CDATA #IMPLIED>
```

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.dtd** zu entscheiden.

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

1. <A>12<C><A>24</C> gültig ungültig
2. <A><C><D att1="10"/></C>24<C><D>12</D></C> gültig ungültig
3. <A><C><D><A>text<D att1="leer"/></D></C> gültig ungültig
4. <A><D></D> gültig ungültig
5. <A><C><D att1="leer"/></C> gültig ungültig
6. <A>12<C><A><C><D><D/></D></C></C> gültig ungültig

(Regeln für Beispiele 1–3: 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:

(9)

Betrachten Sie die folgende Schema-Datei **ns.xsd**:

```
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:t="http://www.dbai.tuwien.ac.at/example"
  targetNamespace="http://www.dbai.tuwien.ac.at/example"
  elementFormDefault="qualified">

  <xs:element name="root-node" type="t:nodeType"/>

  <xs:complexType name="nodeType">
    <xs:sequence minOccurs="0">
      <xs:element name="left-child-node" type="t:nodeType" form="unqualified"/>
      <xs:element name="right-child-node" type="t:nodeType"/>
    </xs:sequence>
    <xs:attribute name="value" type="xs:integer" use="required"/>
    <xs:attribute name="id" type="xs:integer" use="required" form="qualified"/>
    <xs:attribute name="ref" type="xs:integer"/>
  </xs:complexType>
</xs:schema>
```

Kreuzen Sie an, welche Aussagen bzgl. **ns.xsd** (bzw. für ein gültiges Instanzdokument desselben) wahr bzw. falsch sind.

1. Im Instanzdokument kann für das Attribut **id** auch ein anderer Präfix verwendet werden als der im XML Schema deklarierte Präfix **t** wahr falsch
2. Im Instanzdokument kann dem Element **right-child-node** auch ein anderer Namespace als in der XML Schema Datei zugeordnet werden. wahr falsch
3. Das Element **left-child-node** liegt im Instanzdokument im leeren Namespace. wahr falsch
4. Im XML Schema wird bereits deklariert, dass alle Attribute außer **id** keinem Namespace zugeordnet werden müssen. wahr falsch
5. Statt dem Präfix **xs** könnte auch ein anderer Präfix im XML Schema deklariert und verwendet werden. wahr falsch
6. Das Attribut **targetNamespace** dient nur der Dokumentation und kann in der XML Schema Datei auch weggelassen werden. wahr falsch

Aufgabe 3:

(9)

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

1. Wenn ein XML Dokument nicht wohlgeformt ist, muss der Parser die Verarbeitung abbrechen. wahr falsch
2. Der Speicherbedarf eines DOM-Parsers ist unabhängig von der Größe des geparschten XML-Files. wahr falsch
3. Mit einem XMLFilter ist es unter anderem möglich, den Namespace eines Elements zu ändern wahr falsch
4. Das Ergebnis eines XSLT-Stylesheets ist immer ein wohlgeformtes XML Dokument. wahr falsch
5. Namespaces können in DTDs unabhängig vom Präfix definiert werden. wahr falsch
6. CDATA-Sections werden von XML-Parsern ignoriert. wahr falsch

(Regeln für Beispiele 1–3: 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 4 – 7 beziehen sich auf das XML-Dokument **tennis.xml**, das Sie auf der letzten Seite dieser Prüfungsangabe finden.

Aufgabe 4:

(12)

Vervollständigen Sie das XML-Schema Dokument **tennis.xsd**, sodass XML-Dokumente in der Gestalt von **tennis.xml** (siehe Anhang) bezüglich dieses Schemas gültig sind. Berücksichtigen Sie beim Erstellen des Schemas folgende Punkte:

- Vervollständigen Sie die Elemente **viertelfinale**, **halbfinale** und **finale**, sodass sie jeweils vier, zwei und ein **spiel** Element vom komplexen Typ **spielTyp** enthalten müssen.
- Vervollständigen Sie auf der nächsten Seite den komplexen Typ **spielTyp**, sodass ein Spiel entweder aus zwei oder mehr **satz**-Elementen, oder aus einem **wo**-Element bestehen kann. Die Definition der Attribute entnehmen Sie dem XML-Instanzdokument im Anhang.
- Entscheiden Sie selbständig anhand des XML-Instanzdokuments ob Attribute optional oder verpflichtend sind.

Datei **tennis.xsd**:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="turnier">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="viertelfinale">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="spiel" type="spielTyp" minOccurs="4" maxOccurs="4"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="halbfinale">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="spiel" type="spielTyp" minOccurs="2" maxOccurs="2"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="finale">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="spiel" type="spielTyp" minOccurs="1" maxOccurs="1"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
<!-- Fortsetzung auf naechster Seite -->
```

```
<xs:element name="spieler" maxOccurs="unbounded">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="id" type="xs:string" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>

<xs:complexType name="spielTyp">
  <xs:choice>
    <xs:element name="satz" minOccurs="2" maxOccurs="unbounded">
      <xs:complexType>
        <xs:attribute name="erg1" type="xs:integer" use="required"/>
        <xs:attribute name="erg2" type="xs:integer" use="required"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="wo">
      <xs:complexType>
        <xs:attribute name="spieler" type="xs:string" use="required"/>
      </xs:complexType>
    </xs:element>
  </xs:choice>
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="spieler1" type="xs:string" use="required"/>
  <xs:attribute name="spieler2" type="xs:string" use="required"/>
</xs:complexType>
</xs:schema>
```

Aufgabe 5:

(10)

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

- Falls als Ergebnis eine Knotenmenge selektiert wird, geben Sie als Ausgabe die Werte der `id` Attribute an.
- Falls der angegebene XPath Ausdruck keine Knoten selektiert, notieren Sie im entsprechenden Feld "leere Ausgabe".
- Falls als Ergebnis eine Zahl selektiert wird (`count`), geben Sie diese Zahl an.

Betrachten Sie dazu folgendes Beispiel:

```
//viertelfinale/spiel
```

```
g1 g2 g3 g4
```

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

```
/turnier/viertelfinale/spiel[2]
```

```
g2
```

```
//spiel[2]
```

```
g2 g6
```

```
//spiel[4][count(satz) >= 3]
```

```
g4
```

```
//spiel[count(satz) >= 3][4]
```

```
leere Ausgabe
```

```
count(//satz[(../@spieler1 = "p5" and @erg1 > @erg2) or (../@spieler2 = "p5" and @erg2 > @erg1)])
```

```
5
```

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

- Das Wurzelement hat den Namen **resultat** und folgende Kindelemente:
- Für jeden Spieler ein Element mit dessen **id** Attribut als Namen. Dessen Kindelemente sind:
- Jeweils ein Element **spiel** mit Attribut **id** für jedes Spiel, an dem der jeweilige Spieler teilgenommen hat.

Betrachten Sie dazu folgende (auszugsweise) Ausgabe, die ihr XSLT-Stylesheet **tennis.xsl** angewandt auf **tennis.xml** (siehe Anhang) produzieren soll:

```
<resultat>
  <p1>
    <spiel id="g3"/>
  </p1>
  <p2>
    <spiel id="g2"/>
    <spiel id="g5"/>
  </p2>
  ...
```

Vervollständigen Sie hier das XSLT-Stylesheet **tennis.xsl**. Kontrollstrukturen wie z.B. **xsl:for-each** sind für die Lösung grundsätzlich erlaubt, aber nicht erforderlich. Sie brauchen sich nicht um Whitespaces etc. zu kümmern.

Datei **textref.xsl**:

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:template match="/">
    <resultat>
      <xsl:apply-templates select="//spieler"/>
    </resultat>
  </xsl:template>

  <xsl:template match="spieler">
    <xsl:element name="{@id}">
      <xsl:apply-templates select="//spiel[@spieler1=current()/@id or @spieler2=current()/@id]"/>
    </xsl:element>
  </xsl:template>

  <xsl:template match="spiel">
    <spiel id="{@id}"/>
  </xsl:template>

</xsl:stylesheet>
```

Aufgabe 7:

(9)

Betrachten Sie folgende-XQuery Abfrage **tennis.xq**:

```
for $spiel in //halbfinale/spiel
let $ergebnis := $spiel/concat(count(satz[@erg1 >= @erg2]), ":", count(satz[@erg2 >= @erg1]))
let $spieler1 := //spieler[@id = $spiel/@spieler1]/text()
let $spieler2 := //spieler[@id = $spiel/@spieler2]/text()
order by $spieler1, $spieler2, $ergebnis
return
  <spiel>
    <spieler1>{$spieler1}</spieler1>
    <spieler2>{$spieler2}</spieler2>
    <ergebnis>{$ergebnis}</ergebnis>
  </spiel>
```

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

Die exakte Behandlung von Whitespaces ist für diese Beispiel nicht relevant.

```
<spiel>
  <spieler1>Murray</spieler1>
  <spieler2>Nadal</spieler2>
  <ergebnis>2:1</ergebnis>
</spiel>
<spiel>
  <spieler1>Soderling</spieler1>
  <spieler2>Ferrer</spieler2>
  <ergebnis>0:2</ergebnis>
</spiel>
```

Aufgabe 8:

(8)

Vervollständigen Sie die Methode `berechneErgebnis`, welche einem DOM Element `spiel` ein weiteres Attribut `ergebnis` hinzufügt. Dieses Attribut setzt sich aus der Summe der gewonnenen Sätze der Spieler zusammen. Hat zum Beispiel der `spieler1` zwei Sätze gewonnen und `spieler2` nur einen Satz gewonnen steht im `ergebnis`-Attribut der Wert `"2:1"`.

Verwenden Sie bei der Verarbeitung keine XPath Ausdrücke. Um Fehlerbehandlung müssen Sie sich nicht kümmern.

```
public static void berechneErgebnis(Element spiel) {
    int s1Erg = 0;
    int s2Erg = 0;

    NodeList children = spiel.getChildNodes();
    for(int i = 0; i < children.getLength(); i++) {
        if(Node.ELEMENT_NODE == children.item(i).getNodeType()) {
            Element element = (Element) children.item(i);

            if(element.getTagName().equals("satz")) {
                int erg1 = Integer.parseInt(element.getAttribute("erg1"));
                int erg2 = Integer.parseInt(element.getAttribute("erg2"));

                if(erg1 >= erg2) {
                    s1Erg++;
                } else {
                    s2Erg++;
                }
            }
        }
    }
    spiel.setAttribute("ergebnis", s1Erg + ":" + s2Erg);
}
```


Sie können diese Seite abtrennen!

Datei tennis.xml:

```
<turnier>
  <viertelfinale>
    <spiel id="g1" spieler1="p7" spieler2="p3">
      <satz erg1="6" erg2="4"/>
      <satz erg1="6" erg2="7"/>
      <satz erg1="2" erg2="6"/>
    </spiel>
    <spiel id="g2" spieler1="p2" spieler2="p4">
      <satz erg1="7" erg2="6"/>
      <satz erg1="6" erg2="1"/>
    </spiel>
    <spiel id="g3" spieler1="p6" spieler2="p1">
      <wo spieler="p1"/>
    </spiel>
    <spiel id="g4" spieler1="p5" spieler2="p8">
      <satz erg1="1" erg2="6"/>
      <satz erg1="6" erg2="2"/>
      <satz erg1="7" erg2="6"/>
    </spiel>
  </viertelfinale>

  <halbfinale>
    <spiel id="g5" spieler1="p3" spieler2="p2">
      <satz erg1="3" erg2="6"/>
      <satz erg1="7" erg2="6"/>
      <satz erg1="6" erg2="0"/>
    </spiel>
    <spiel id="g6" spieler1="p6" spieler2="p5">
      <satz erg1="3" erg2="6"/>
      <satz erg1="6" erg2="7"/>
    </spiel>
  </halbfinale>

  <finale>
    <spiel id="g7" spieler1="p3" spieler2="p5">
      <satz erg1="4" erg2="6"/>
      <satz erg1="7" erg2="6"/>
      <satz erg1="6" erg2="4"/>
    </spiel>
  </finale>

  <spieler id="p1">Djokovic</spieler>
  <spieler id="p2">Nadal</spieler>
  <spieler id="p3">Murray</spieler>
  <spieler id="p4">Federer</spieler>
  <spieler id="p5">Ferrer</spieler>
  <spieler id="p6">Soderling</spieler>
  <spieler id="p7">Berdych</spieler>
  <spieler id="p8">Fish</spieler>
</turnier>
```

Gesamtpunkte: 75