

Gruppe A	PRÜFUNG AUS "SEMISTRUKTURIERTE DATEN" 181.135		01. 12. 2010
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 Studierendenausweis bereit.

### Aufgabe 1:

(9)

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

```
<!ELEMENT A (#PCDATA|B|C|D)*>
<!ELEMENT B (A|C|D)*>
<!ELEMENT C (#PCDATA)>
<!ELEMENT D (#PCDATA)>
<!ATTLIST C E CDATA #IMPLIED>
<!ATTLIST D E CDATA #REQUIRED>
```

Betrachten Sie weiters die sechs verschiedenen XML-Dateien, die unten angeführt sind.

Hinweise:

- Gehen Sie davon aus, dass allen folgenden Dateien die Zeilen  

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE A SYSTEM "test.dtd">
```

 vorangestellt sind.
- Sie können auch 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><B>abc</B></A>               | gültig <input type="radio"/>            | ungültig <input checked="" type="radio"/> |
| 2. <A><B><A>abc</A></B></A>        | gültig <input checked="" type="radio"/> | ungültig <input type="radio"/>            |
| 3. <A><B><B></B></B></A>           | gültig <input type="radio"/>            | ungültig <input checked="" type="radio"/> |
| 4. <A><C E="abc"/>abc</A>          | gültig <input checked="" type="radio"/> | ungültig <input type="radio"/>            |
| 5. <A><B><D E="abc"/>abc</B></A>   | gültig <input type="radio"/>            | ungültig <input checked="" type="radio"/> |
| 6. <A><B><A><D>abc</D></A></B></A> | gültig <input type="radio"/>            | ungültig <input checked="" type="radio"/> |

(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:

(6)

Betrachten Sie die folgende XML- Datei **ns.xml**:

```
<A xmlns="namespace1">
  <ns:B xmlns:ns="namespace2">
    <D>abc</D>
  </ns:B>
  <C xmlns="namespace2">
    <E/>
  </C>
</A>
```

Kreuzen Sie an, ob die folgenden Aussagen für die Datei **ns.xml** wahr oder falsch sind.

1. Element B liegt im Namespace "namespace2". wahr  falsch
2. Element D liegt im Namespace "namespace2". wahr  falsch
3. Element C liegt im Namespace "namespace1". wahr  falsch
4. Element E liegt im Namespace "namespace1". wahr  falsch

(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 3:

(12)

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

1. Bei SAX-Parsern muss sich im Allgemeinen der Programmierer / die Programmiererin um die Überprüfung auf Wohlgeformtheit des eingelesenen XML-Dokuments kümmern. wahr  falsch
2. In einem wohlgeformten XML-Dokument müssen die lokalen Namen der Attribute innerhalb eines Elementstarttags verschieden sein. wahr  falsch
3. Die DTD Elementdeklaration (B+,C,C\*) ist gleichwertig mit (B,B\*,C+). wahr  falsch
4. Rekursive Definitionen wie z.B. <!ELEMENT B (B?,C)> sind in DTDs erlaubt. wahr  falsch
5. DOM erlaubt wahlfreien Zugriff auf das gesamte XML Dokument. wahr  falsch
6. SAX: Textevents können auch direkt hintereinander auftreten. wahr  falsch
7. Die Ordnung der Knoten im Resultat eines XPath-Ausdrucks ist immer in Document Order. wahr  falsch
8. Der XPath-Ausdruck //status ist die Kurzschreibweise des XPath-Ausdrucks /descendant::status wahr  falsch

(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)

Die folgenden Aufgaben 4 – 8 beziehen sich auf das XML-Dokument `bbs.xml`, das Sie auf der letzten Seite dieser Prüfungsangabe finden.

**Aufgabe 4:**

(10)

Beachten Sie folgende XPath-Abfragen angewandt auf das Dokument `bbs.xml` an (siehe Anhang).

Beachten Sie dazu folgendes Beispiel:

```
//user/@kuerzel | //message/@id
```

```
SWO ESA MPI MMO 1 2 3 4 5 6
```

Geben Sie nun die Ausgaben der folgenden XPath Ausdrücke an.

```
//message[not(message)]/@id
```

```
2 3 6
```

```
//user[@kuerzel=//link/@refer]/@kuerzel
```

```
ESA MPI MMO
```

```
//message[position()=1]/@id
```

```
1 2 5 6
```

```
//user[contains(@kuerzel, 'M')]/@kuerzel
```

```
MPI MMO
```

```
//message[@id=3]/preceding-sibling::message/@id
```

```
2
```

### Aufgabe 5:

(12)

Vervollständigen Sie das XML Schema **bbs.xsd**, sodass XML-Dokumente in der Gestalt von **bbs.xml** (siehe Anhang) bezüglich dieses XML Schemas gültig sind. Die XML-Dokumente speichern User und Messages in einem Messageboard-System. Berücksichtigen Sie beim Erstellen des XML Schemas folgende Punkte:

- Wählen Sie für die Attribute entsprechende Typen aus. Versuchen Sie aus dem Beispieldokument **bbs.xml** herauszulesen, welche Attribute verpflichtend sind, und welche nicht.
- Beachten Sie, dass `message` Elemente gemischten Inhalt haben.
- Die Rekursion vom `message` Elementen soll uneingeschränkt möglich sein.
- Geben Sie auch die entsprechenden Schlüsseldefinition zwischen
  - dem `refer` Attribut des Elements `link` und dem `kuerzel` Attribut von `user` sowie
  - dem `user` Attribut des Elements `message` und dem `kuerzel` Attribut von `user`.

Verwenden Sie hierfür `key` and `keyref`.

Vervollständigen sie im folgenden Dokument **bbs.xsd** die Definition des Elements `bbs` und definieren Sie den Typ `mtype`.

Datei **bbs.xsd**:

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

  <xsd:element name="bbs">

    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="user" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:attribute name="kuerzel" type="xsd:string" use="required"/>
            <xsd:attribute name="name" type="xsd:string" use="required"/>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="message" type="mtype" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>

  </xsd:element>

</xsd:schema>
```

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

```
<xsd:key name="primary">  
  <xsd:selector xpath="user"/>  
  <xsd:field xpath="@kuerzel"/>  
</xsd:key>
```

```
<xsd:keyref name="foreign_message_head" refer="primary">  
  <xsd:selector xpath="//message"/>  
  <xsd:field xpath="@user"/>  
</xsd:keyref>
```

```
<xsd:keyref name="foreign_message_text" refer="primary">  
  <xsd:selector xpath="//message/link"/>  
  <xsd:field xpath="@refer"/>  
</xsd:keyref>
```

```
</xsd:element>
```

```
<xsd:complexType name="mtype" mixed="true">  
  <xsd:choice minOccurs="0" maxOccurs="unbounded">  
    <xsd:element name="message" type="mtype"/>  
    <xsd:element name="link">  
      <xsd:complexType>  
        <xsd:attribute name="refer" type="xsd:string" use="required"/>  
      </xsd:complexType>  
    </xsd:element>  
  </xsd:choice>  
  <xsd:attribute name="date" type="xsd:date" use="required"/>  
  <xsd:attribute name="user" type="xsd:string" use="required"/>  
  <xsd:attribute name="id" type="xsd:integer" use="required"/>  
  <xsd:attribute name="titel" type="xsd:string"/>  
</xsd:complexType>
```

```
</xsd:schema>
```

### Aufgabe 6:

(8)

Erstellen Sie eine XQuery-Anfrage **xquery.xq**, die angewandt auf Dokumente der Gestalt **bbs.xml** folgende Ausgabe liefert:

- Für jedes Element **user** soll die Anzahl der geposteten Messages (**message** Elemente, die als **user** Attribut das entsprechende Kürzel enthalten) berechnet werden.
- Es sollen nur jene **user** Elemente ausgegeben werden, die
  - mehr als drei Messages gepostet haben **oder**
  - verlinkt wurden (d.h. deren Kürzel in einem **link** Element als **refer** Attribut vorkommt), oder beides
- Die Ausgabe soll in folgender Form erfolgen:  
`<messages name='name '>anzahl </messages>`

Beispielsweise ist die gewünschte Ausgabe, angewandt auf **bbs.xml**:

```
<messages name="Stefan Woltran">3</messages>
<messages name="Emanuel Sallinger">1</messages>
<messages name="Markus Pichlmair">1</messages>
<messages name="Michael Morak">1</messages>
```

Geben Sie hier die XQuery-Anfrage **xquery.xq** an:

Datei **xquery.xq**:

```
for $user in //user
let $count := count(//message[@user=$user/@kuerzel])
where $count > 2 or //link[@refer=$user/@kuerzel]
return <messages name="{ $user/@name }">{ $count }</messages>
```

### Aufgabe 7:

(10)

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

- **message** Elemente sollen in der Ausgabe
  - in **post** umbenannt werden
- **link** Elemente sollen in der Ausgabe
  - durch den Namen (Attribut **name**) des entsprechenden **user** Elements ersetzt werden
- Ansonsten soll das Dokument unverändert in die Ausgabe übernommen werden

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

Datei **xslt.xml**:

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

  <xsl:template match="message">
    <post>
      <xsl:apply-templates select="@*|node()"/>
    </post>
  </xsl:template>

  <xsl:template match="link">
    <xsl:value-of select="//user[@kuerzel=current()/@refer]/@name"/>
  </xsl:template>

  <xsl:template match="@*|node()">
    <xsl:copy>
      <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
  </xsl:template>

</xsl:stylesheet>
```

### Aufgabe 8:

(8)

Vervollständigen Sie die Methode `editMessage`, die angewandt auf das DOM Element `m` (ein `message` Element aus einem Dokument der Form `bbs.xml`) folgende Änderung durchführt: Das `date` Attribut wird durch ein Kindelement ersetzt. D.h.

- Fügen Sie ein Kindelement `date` hinzu, das den Wert des `date` Attributs als Textinhalt enthält
- Entfernen Sie das `date` Attribut

Um Fehlerbehandlung müssen Sie sich nicht kümmern. Insbesondere können Sie annehmen, dass `m` tatsächlich ein `date` Attribut besitzt. Weiters ist die Reihenfolge der Kindelemente nicht relevant.

```
public static void editMessage(Element m) {
    Attr dateAttr = m.getAttributeNode("date");
    Element dateEl = m.getOwnerDocument().createElement("date");
    dateEl.setNodeValue(dateAttr.getNodeValue());
    m.removeAttributeNode(dateAttr);
    m.appendChild(dateEl);
}
```

Sie können diese Seite abtrennen!

Datei bbs.xml:

```
<bbs>
  <user kuerzel="SW0" name="Stefan Woltran"/>
  <user kuerzel="ESA" name="Emanuel Sallinger"/>
  <user kuerzel="MPI" name="Markus Pichlmair"/>
  <user kuerzel="MMO" name="Michael Morak"/>
  <!-- .... -->
  <message date="2010-11-24" user="SW0" titel="Pruefungsangabe" id="1">
    Pruefungsangabe ist fertig. <link refer="ESA"/> und <link refer="MMO"/>,
    schaut Euch bitte die Angabe an.
    <message date="2010-11-26" user="ESA" id="2">
      Ja, schaut gut aus!
    </message>
    <message date="2010-11-26" user="MMO" id="3">
      Aus meiner Sicht auch ok.
    </message>
  </message>
  <message date="2010-11-30" user="SW0" titel="Pruefungsaufsicht" id="4">
    Liebe Kollegen, kann morgen leider nicht Aufsicht machen...
    <message date="2010-11-30" user="MPI" titel="RE:Pruefungsaufsicht" id="5">
      Ich kann einspringen...
      <message date="2010-11-30" user="SW0" id="6">
        Danke, <link refer="MPI"/>.
      </message>
    </message>
  </message>
</bbs>
```

Gesamtpunkte: 75