

Gruppe A	PRÜFUNG AUS "SEMISTRUKTURIERTE DATEN" 184.705		2. 12. 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 (#PCDATA | B)*>
```

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 checked="" type="checkbox"/> | ungültig <input type="checkbox"/> |
| 2. <A><C>a</C> | gültig <input type="checkbox"/> | ungültig <input checked="" type="checkbox"/> |
| 3. <A><C>a</C><A/> | gültig <input checked="" type="checkbox"/> | ungültig <input type="checkbox"/> |
| 4. <A><A/><A/> | gültig <input type="checkbox"/> | ungültig <input checked="" type="checkbox"/> |
| 5. <A><A/><A/> | gültig <input checked="" type="checkbox"/> | ungültig <input type="checkbox"/> |
| 6. <A><C>ab<C>c</C></C><A/> | gültig <input checked="" type="checkbox"/> | ungültig <input type="checkbox"/> |
| 7. <A>a<C>bc</C><A/> | gültig <input type="checkbox"/> | ungültig <input checked="" type="checkbox"/> |
| 8. <A><C/>a<A/> | gültig <input type="checkbox"/> | ungültig <input checked="" type="checkbox"/> |

(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. Pro XML-Dokument ist maximal eine CDATA-Section erlaubt. wahr falsch
2. Der Inhalt einer CDATA-Section wird als normaler Text an den Parser weitergeleitet, auch wenn darin Tags vorkommen. wahr falsch
3. Um die Wohlgeformtheit eines XML-Dokuments zu überprüfen, muss eine DTD angegeben werden. wahr falsch
4. Um die Wohlgeformtheit eines XML-Dokuments zu überprüfen, muss ein XML Schema angegeben werden. wahr falsch
5. XSLT ist eine W3C-Recommendation. wahr falsch
6. SAX Parser ignorieren Attributwerte. wahr falsch
7. Der Speicherbedarf eines DOM-Parsers ist im Allgemeinen geringer als eines SAX-Parsers wahr falsch
8. In XPath gilt $(1,2) \neq (2,3)$ wahr falsch
9. Rekursive Definitionen wie z.B. `<!ELEMENT B (A,B?,C)>` sind in DTDs erlaubt. wahr falsch
10. Der XPath-Ausdruck `//subpage` ist die Kurzschreibweise des XPath-Ausdrucks `/descendant::subpage` 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 `browser.xml`, das Sie auf der letzten Seite dieser Prüfungsangabe finden.

Aufgabe 3:

(12)

Vervollständigen Sie die XML Schema Datei `browser.xsd`, sodass XML-Dokumente in der Gestalt von `browser.xml` (siehe Anhang) bezüglich dieses Schemas gültig sind. XML-Dokumente der Form `browser.xml` sollen den aktuellen Zustand eines fiktiven Webbrowsers abspeichern. Berücksichtigen Sie beim Erstellen des Schemas folgende Punkte:

- Das Element `browser` ist das Wurzelement und besteht aus mindestens einem `tab`-Element.
- Jedes `tab`-Element beinhaltet Information zu einer HTML-Seite. Eine solche Seite besteht aus genau einem `content`-Element gefolgt von beliebig vielen (beliebig geordneten) `subpage`- und `sublink`-Elementen.
- `sublink` ist ein leeres Element mit einem Attribut `url`; `subpage` beinhaltet Information zu einer HTML-Seite (siehe oben).
- Wählen Sie anhand des Dokuments im Anhang sinnvolle Typen und Häufigkeiten aus. Bilden Sie die vorhandene Rekursion durch einen geeigneten selbst-definierten Typ ab.
- Alle Attribute sollen verpflichtend sein. Dies soll in Ihrer XML-Schema Definition explizit ablesbar sein. Es sind keine Schlüssel zu definieren.

Datei `browser.xsd`:

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

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

  <xsd:element name="browser">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="tab" maxOccurs="unbounded" type="page"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:complexType name="page">
    <xsd:sequence>
      <xsd:element name="content" type="xsd:string"/>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="sublink">
          <xsd:complexType>
            <xsd:attribute name="url" type="xsd:string"/>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="subpage" type="page"/>
      </xsd:choice>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:string"/>
  </xsd:complexType>
```

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

Aufgabe 4:

(10)

Betrachten Sie die folgenden XPath-Abfragen angewandt auf das Dokument **browser.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 der angegebene XPath Ausdruck ein Element **subpage** selektiert, geben Sie den Wert des Attributs **id** an.
- Falls der angegebene XPath Ausdruck ein Element **sublink** selektiert, geben Sie den Wert des Attributs **url** zurück. Es ist ausreichend, die letzten drei Zeichen der URL anzugeben.

Betrachten Sie dazu folgendes Beispiel:

```
//subpage
```

```
s1 s2 s3 s4
```

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

```
//subpage[@id='s3']/..
```

```
s2
```

```
//subpage[not(sublink)]
```

```
s1 s3 s4
```

```
//subpage[subpage]
```

```
s2
```

```
//sublink[2]
```

```
http://en.wikipedia.org/wiki/CDATA          http://en.wikipedia.org/wiki/XML
```

```
(//sublink)[2]
```

```
http://en.wikipedia.org/wiki/CDATA
```

Aufgabe 5:

(8)

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

```
for $s in //subpage
let $i := $s/@id
let $c := count($s//sublink)
let $a := count($s//ancestor::subpage)
return element {$i}{$c,$a}
```

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

```
<s1>0 1</s1>
<s2>3 2</s2>
<s3>0 2</s3>
<s4>0 1</s4>
```

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

- Es soll ein Element **liste** erzeugt werden, das für das zweite **tab** Element folgende Ausgabe enthält:
- Es soll ein Element **anzahl** erzeugt werden, das die Anzahl der direkt als Kindelemente dieses Tabs enthaltenen **sublink** Elemente enthält.
- Für jedes in diesem Tab enthaltene **sublink** Element (sowohl als direkte Kindelemente, als auch tiefer verschachtelt) soll ein Element **link** erzeugt werden, dessen Textinhalt der Wert des **url** Attributs ist.

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

```
<liste>
  <anzahl>0</anzahl>
  <link>http://en.wikipedia.org/wiki/HTML</link>
  <link>http://en.wikipedia.org/wiki/XML</link>
  <link>folgt</link>
</liste>
```

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

Datei **browser.xsl**:

```
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <liste>
      <xsl:apply-templates select="browser/tab[2]" />
    </liste>
  </xsl:template>

  <xsl:template match="tab">
    <anzahl><xsl:value-of select="count(sublink)"/></anzahl>
    <xsl:apply-templates/>
  </xsl:template>

  <xsl:template match="sublink">
    <link><xsl:value-of select="@url"/></link>
  </xsl:template>

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

Aufgabe 7:

(8)

Vervollständigen Sie den folgenden SAX Handler, der angewandt auf Dokumente der Gestalt **browser.xml** den Inhalt jedes **content**-Elements ausgibt, das innerhalb eines **subpage**-Elements vorkommt.

Für das Dokument **browser.xml** wird beispielsweise folgender Text ausgegeben:

```
here we have some cached html ...
and some nested subpages ...
<b>here is some bold <i>italic</i> text </b>
a final one...
```

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

```
public class CheckTotal extends DefaultHandler {

    int subpageLevel = 0;
    boolean inContent = false;

    public void startElement(String uri, String localName, String qName, Attributes atts)
        throws SAXException {
        if ("subpage".equals(localName)) {
            subpageLevel++;
        }
        if ("content".equals(localName)) {
            inContent = true;
        }
    }

    public void endElement(String uri, String localName, String qName) throws SAXException {
        if ("subpage".equals(localName)) {
            subpageLevel--;
        }
        if ("content".equals(localName)) {
            inContent = false;
        }
    }

    public void characters(char[] ch, int start, int length) throws SAXException {
        if (subpageLevel > 0 && inContent) {
            for (int i = start; i <= length; i++) {
                System.out.print(ch[i]);
            }
        }
    }
}
```

Sie können diese Seite abtrennen!

Datei browser.xml:

```
<browser>
  <tab id="t1">
    <content>
      <![CDATA[ <head>here is some (bad) html <br> </head> ]]>
    </content>
    <sublink url="http://de.wikipedia.org/wiki/W3C"/>
    <subpage id="s1">
      <content>
        here we have some cached html ...
      </content>
    </subpage>
    <sublink url="http://en.wikipedia.org/wiki/CDATA"/>
  </tab>
  <tab id="t2">
    <content>
      here we have some html again ...
    </content>
    <subpage id="s2">
      <content>
        and some nested subpages ...
      </content>
      <subpage id="s3">
        <content>
          <![CDATA[ <b>here is some bold <i>italic</i> text </b> ]]>
        </content>
      </subpage>
      <sublink url="http://en.wikipedia.org/wiki/HTML"/>
      <sublink url="http://en.wikipedia.org/wiki/XML"/>
      <sublink url="folgt"/>
    </subpage>
    <subpage id="s4">
      <content>
        a final one ...
      </content>
    </subpage>
  </tab>
</browser>
```

Gesamtpunkte: 75