PRÜFUNG IN "SEMI-STRUKTURIERTE DATEN" **184.705**			27. 11. 2017
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 folgende XML Schema Datei **test.xsd**:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="A">
        <xsd:complexType mixed="true" >
            <xsd:choice maxOccurs="2">
                <xsd:element name="B" type="xsd:integer"/>
                <xsd:element name="C" type="typeC">
                    <xsd:key name="constraint1">
                        <xsd:selector xpath="D"/>
                        <xsd:field xpath="@id"/>
                    </xsd:key>
                </xsd:element>
            </xsd:choice>
        </xsd:complexType>
    </xsd:element>

    <xsd:complexType name="typeC" mixed="false">
         <xsd:sequence>
                <xsd:element name="D" type="typeD" minOccurs="0" maxOccurs="10"/>
          </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="typeD" mixed="false">
        <xsd:attribute name="id" type="xsd:integer"/>
    </xsd:complexType>
 </xsd:schema>
```

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.xsdzu entscheiden.

Kreuzen Sie an, welche der folgenden xml-Dateien gültig bezüglich test.xsd sind.

1. `<A><B>1</B><B>23</B></A>` valid ⊗ invalid ○
2. `<A><C><D/></C><B>1</B></A>` valid ○ invalid ⊗
3. `<A><B>1</B> dbai <C/></A>` valid ⊗ invalid ○
4. `<A><C><D id="10"/><D id="11"/><D id="10"/></C></A>` valid ○ invalid ⊗
5. `<A><C><D id="10"/><D id="11"/></C><C><D id="10"/><D id="11"/></C></A>` valid ⊗ invalid ○
6. `<A><B>12</B><B>24</B><C/></A>` valid ○ invalid ⊗
7. `<A>Donald Duck</A>` valid ○ invalid ⊗
8. `<A><C/></A>` valid ⊗ invalid ○

(Für jede korrekte Antwort 1.5 Punkte, **für jede falsche Antwort -1.5 Punkte**, unbeanwortete Fragen 0 Punkt, Insgesamt nicht weniger als 0 Punkte)

Aufgabe 2: (15)

Entscheiden Sie, ob die folgenden Aussagen wahr oder falsch sind.

1. Event-based Parser benötigen nur konstant viel Speicher, d.h. der Speicherbedarf ist unabhängig von der Größe des geparsten XML Dokuments. wahr ⊗ falsch ◯
2. SAX ist ein Standard für Tree-Based XML-Parser. wahr ◯ falsch ⊗
3. XPath ist Teil der XQuery Sprache. wahr ⊗ falsch ◯
4. XSLT ist Teil der XPath Sprache. wahr ◯ falsch ⊗
5. Das Wurzelelement eines XML Dokuments darf keine Attribute haben. wahr ◯ falsch ⊗
6. Namespace URIs sind immer gültige XML Namen. wahr ◯ falsch ⊗
7. Attribute ohne Prefix sind im Default Namespace. wahr ◯ falsch ⊗
8. Im `match`-Attribut eines `<xsl:template>`-Elements eines XSLT-Stylesheets dürfen beliebige XPath-Ausdrücke verwendet werden. wahr ◯ falsch ⊗
9. DTD steht für Data Type Definition. wahr ◯ falsch ⊗
10. XML Schemas erlauben typischerweise eine exaktere Spezifikation als DTDs. wahr ⊗ falsch ◯

(Für jede korrekte Antwort 1.5 Punkte, **für jede falsche Antwort -1.5 Punkte**, unbeanwortete Fragen 0 Punkt, Insgesamt nicht weniger als 0 Punkte)

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

Aufgabe 3: (12)

Vervollständigen Sie das DTD Dokument **bikerental.dtd**, sodass XML-Dokumente in der Gestalt von **bikerental.xml** (siehe Anhang) bezüglich dieser DTD gültig sind. Berücksichtigen Sie beim Erstellen der DTD folgende Punkte:

- Das `vehicles` Element enthält beliebig viele `vehicle` Elemente. Das `employees` Element enthält beliebig viele `mechanic` und `salesperson` Elemente in beliebiger Reihenfolge. Das `reservations` Element enthält beliebig viele `reservation` Elemente.
- `vehicle` Elemente enthalten ein `type` Element, ein `available` Element, möglicherweise ein `capacity` Element und immer ein `price` Element (in dieser Reihenfolge).
- `mechanic` und `salesperson` Elemente enthalten ein `name` Element und `mechanic` Elemente noch mindestens ein `repairs` Element.
- `vehicle, mechanic` und `salesperson` Elemente haben als Attribut eine eindeutige ID.
- `reservation` Elemente enthalten ein `customer` Element gefolgt von mindestens einem `resere` Element.
- Das Attribut `vehicleid` verweist immer auf die ID eines `vehicle` Elements. Das Attribut `responsible_employee` verweist immer auf die ID eines Mitarbeiters.
- Wenn nicht angegeben treffen Sie plausible Annahmen über Typen von Attributen und Elementen.

File **bikerental.dtd**:

```
<!ELEMENT bikerental (vehicles, employees, reservations)>

<!ELEMENT vehicles (vehicle*)>
<!ELEMENT employees (salesperson|mechanic)*>
<!ELEMENT reservations (reservation*)>

<!ELEMENT vehicle (type,available,capacity?,price)>
<!ATTLIST vehicle id ID #REQUIRED>

<!ELEMENT salesperson (name)>
<!ATTLIST salesperson id ID #REQUIRED>

<!ELEMENT mechanic (name, repairs+)>
<!ATTLIST mechanic id ID #REQUIRED>

<!ELEMENT repairs EMPTY>
<!ATTLIST repairs vehicleid IDREF #REQUIRED>

<!ELEMENT reservation (customer,reserve+)>
<!ATTLIST reservation responsible_employee IDREF #REQUIRED time CDATA #REQUIRED>

<!ELEMENT reserve EMPTY>
<!ATTLIST reserve vehicleid IDREF #REQUIRED quantity CDATA #REQUIRED>


<!ELEMENT type (#PCDATA)>
<!ELEMENT available (#PCDATA)>
<!ELEMENT capacity (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT customer (#PCDATA)>
```

Aufgabe 4: (10)

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

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

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

```
count(//reserve[@quantity<3])
```

```
3
```

```
//vehicle[@id=//reservation/reserve/@vehicleid][2]/type
```

```
<type>conference_bike</type>
```

```
/bikerental//vehicle[last()][capacity]/type
```

```
leere Ausgabe
```

```
//vehicle[capacity][last()]/type/text()
```

```
inline_skates
```

```
//vehicle[not(@id=//mechanic/repairs/@vehicleid)][2]/type
```

```
<type>tandem</type>
```

Aufgabe 5: (8)

Betrachten Sie folgende XQuery Abfrage **xquery.xq**:

```
<customers>
{
for $c in distinct-values(//customer/text())
order by $c
return
   <customer id="{$c}">
   {
      for $r in distinct-values(//reservation[customer/text()=$c]/reserve/@vehicleid)
      let $v := //vehicle[@id=$r]
      order by $v/capacity, $v/price
      return
         <rented>{$v/type/text()}</rented>
   }
   </customer>
}
</customers>
```

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

Sie müssen sich nicht um Whitespaces kümmern.

```
<?xml version="1.0" encoding="UTF-8"?>
<customers>
    <customer id="DBAI">
        <rented>inline_skates</rented>
        <rented>bicycle</rented>
        <rented>conference_bike</rented>
    </customer>
    <customer id="Yasemin">
        <rented>bicycle</rented>
    </customer>
</customers>
```

Aufgabe 6: (12)

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

- Ausgegeben werden soll ein XML Dokument mit dem Wurzelelement `large_bikes`.
- Es soll für jedes `vehicle` Element mit einer `capacity` von mehr als 5 ein Element mit dem Fahrzeug `type` als Namen ausgegeben werden.
- Zusätzlich sollen alle Reservierungen des Fahrzeugs folgendermaßen ausgegeben werden:
  - Pro Reservierung wird ein `reserved` Element erstellt.
  - Jedes `reserved` Element hat ein `customer` und ein `time` Element als Kinder. In diesen soll der `customer` der Reservierung und die `time` der Reservierung ausgegeben werden.

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

```
<?xml version="1.0" encoding="UTF-8"?>
<large_bikes>
    <conference_bike>
        <reserved>
            <customer>DBAI</customer>
            <time>24/07/2018 17:00</time>
        </reserved>
        <reserved>
            <customer>DBAI</customer>
            <time>26/11/2018 13:00</time>
        </reserved>
    </conference_bike>
</large_bikes>
```

Vervollständigen Sie hier das XSLT-Stylesheet **textref.xsl**. Sie brauchen sich nicht um Whitespaces etc. zu kümmern.

File **bikerental.xsl**:

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

    <xsl:template match="bikerental">
        <large_bikes>
        <xsl:apply-templates select="//vehicle[capacity/text()&gt;5]" />
        </large_bikes>
    </xsl:template>

    <xsl:template match="vehicle">
        <xsl:element name="{type}">
           <xsl:apply-templates select="//reserve[@vehicleid=current()/@id]" />
        </xsl:element>
    </xsl:template>

    <xsl:template match="reserve">
        <reserved>
            <customer><xsl:value-of select="../customer" /></customer>
            <time><xsl:value-of select="../@time" /></time>
        </reserved>
    </xsl:template>

</xsl:stylesheet>
```

Aufgabe 7: (6)

Betrachten Sie die folgende Java Klasse und geben Sie die Ausgabe der Klasse an, wenn als Input die Datei **bikerental.xml** verwendet wird.

```
public class RunSAX extends DefaultHandler {
    private String eleText, id, t, c;
    private int total = 0;
    private HashMap<String, Integer> v = new HashMap<String, Integer>();

    @Override
      public void characters(char[] text, int start, int length)
                  throws SAXException {
          eleText = new String(text, start, length);
      }

    @Override
    public void startElement(String namespaceURI, String localName, String qName, Attributes atts)
     throws SAXException {
        if ("vehicle".equals(localName))
            id = atts.getValue("id");

        if ("reservation".equals(localName))
            t = atts.getValue("time");

        if ("reserve".equals(localName))
            total += v.get(atts.getValue("vehicleid")) * Integer.parseInt(atts.getValue("quantity"));
    }

    @Override
    public void endElement(String namespaceURI, String localName, String qName) throws SAXException {
        if ("price".equals(localName))
            v.put(id, Integer.parseInt(eleText));

        if ("customer".equals(localName))
            c = eleText;

        if ("reservation".equals(localName)) {
            System.out.println(c + " (" + t + "): " + total);
            total = 0;
        }
    }

    public static void main(String[] args) throws Exception {
        if (args.length != 1) {
            System.err.println("Usage: java RunSAX <input.xml>");
            System.exit(1);
        }

        String input = args[0];
        InputSource source = new InputSource(new FileInputStream(input));

        XMLReader xr = XMLReaderFactory.createXMLReader();
        RunSAX rs = new RunSAX();
        xr.setContentHandler(rs);

        xr.parse(source);
    }
}
```

Geben Sie hier die Ausgabe an:

```
DBAI (24/07/2018 17:00): 170
Yasemin (11/5/2018 18:00): 40
DBAI (26/11/2018 13:00): 180
```

Total points: 75

File **bikerental.xml**: **You may separate this page!**

```
<bikerental>
  <vehicles>
      <vehicle id="v1">
          <type>bicycle</type>
          <available>20</available>
          <capacity>1</capacity>
          <price>20</price>
      </vehicle>
      <vehicle id="v2">
          <type>tandem</type>
          <available>2</available>
          <capacity>2</capacity>
         <price>18</price>
      </vehicle>
      <vehicle id="v3">
          <type>conference_bike</type>
          <available>5</available>
          <capacity>7</capacity>
         <price>60</price>
      </vehicle>
      <vehicle id="v4">
          <type>inline_skates</type>
          <available>10</available>
          <capacity>1</capacity>
         <price>10</price>
     </vehicle>
      <vehicle id="v5">
          <type>roller_skates</type>
          <available>10</available>
          <price>10</price>
      </vehicle>
  </vehicles>
  <employees>
      <mechanic id="e1">
          <name>Thomas</name>
          <repairs vehicleid="v4"/>
          <repairs vehicleid="v5"/>
      </mechanic>
      <salesperson id="e2">
          <name>Travelling</name>
      </salesperson>
  </employees>
  <reservations>
    <reservation time="24/07/2018 17:00" responsible_employee="e2">
        <customer>DBAI</customer>
        <reserve vehicleid="v1" quantity="5"/>
        <reserve vehicleid="v3" quantity="1"/>
        <reserve vehicleid="v4" quantity="1"/>
    </reservation>
    <reservation time="11/5/2018 18:00" responsible_employee="e1">
        <customer>Yasemin</customer>
        <reserve vehicleid="v1" quantity="2"/>
    </reservation>
    <reservation time="26/11/2018 13:00" responsible_employee="e2">
        <customer>DBAI</customer>
        <reserve vehicleid="v3" quantity="3"/>
    </reservation>
  </reservations>
</bikerental>
```