PRÜFUNG IN "SEMI-STRUKTURIERTE DATEN" **184.705**			8. 1. 2019
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**:

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <xsd:element name="A" type="Atype"/>

    <xsd:complexType name="Atype">
        <xsd:choice minOccurs="0" maxOccurs="2">
            <xsd:element name="D" type="xsd:int" minOccurs="0" maxOccurs="1"/>
            <xsd:element name="B" type="Btype" minOccurs="0" maxOccurs="1"/>
        </xsd:choice>
    </xsd:complexType>

    <xsd:element name="B" type="Btype"/>

    <xsd:complexType name="Btype">
        <xsd:sequence>
            <xsd:element name="A" type="Atype"/>
            <xsd:element name="C" type="xsd:int" minOccurs="0" maxOccurs="2"/>
        </xsd:sequence>
    </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.xsd zu entscheiden.

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

1. `<A/>` valid ○ invalid ○
2. `<A><B><A/></B><D>2</D></A>` valid ○ invalid ○
3. `<B><A/></B>` valid ○ invalid ○
4. `<A><D>-2147483648</D><D>34</D></A>` valid ○ invalid ○
5. `<A><D>2147483647</D><B><A><B><A/></B></A></B></A>` valid ○ invalid ○
6. `<A><D>42</D><B><A/><C>21</C><C>21</C></B></A>` valid ○ invalid ○
7. `<B><A><B><C>21</C></B></A><C>21</C><C>21</C></B>` valid ○ invalid ○
8. `<A><D>1</D><D>2</D><B><A/></B><B><A/></B></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)
Beantworten Sie, die folgenden Fragen kurz und bündig (Für jede korrekte Antwort 1.5 Punkte).

1. Geben Sie zwei Formalismen (zwei Formate) an um semistrukturierte Daten zu speichern:

2. Wozu benötigt man die in XML vordefinieren entity references?

3. Geben Sie ein Beispiel für eine Document Type Declaration.

4. Geben Sie 5 der möglichen Attribut Typen in DTDs an.

5. Welche Einschränkung bezüglich der Achsen gibt es für XPath Ausdrücke in XML-Schema Definitionen?

6. Was macht das Default-Template in XSLT für Attribute?

7. Definieren sie kurz die vier verschiedenen Arten von Inhalt (content) die ein xml-Element haben kann. (6)

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

Aufgabe 3: (12)

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

- Das Wurzel Element hat zuerst ein `address` Element dann ein `description` Element und dann beliebig viele `user` und `room` Elemente als Kinder.
- Das `description` Element enthält eine textuelle Beschreibung und kann Referenzen mittels `ref` Elementen enthalten. Ein `ref` Element verweist entweder auf einen Raum und enthält ein Attribut `room` oder auf einen user und enthält ein Attribut `user`.
- In `user` Element enthält wird der Name entweder als `fullname` oder als `uname` Element gespeichert. Ein `fullname` Element enthält den Namen als Text. Ein `uname` Element enthält ein `firstname` Element das den Vornamen speichert und ein `lastname` Element das den Nachnamen speichert.
- Jedes `room` Element hat eine eindeutige ID `id` und kann einem Nutzer mittels `owner` Attribut zugeordnet werden. Desweitern enthält es ein `name` Element, ein `temperature` Element das die aktuelle Raumtemperatur angibt, ein `radiator_level` Element das die aktuelle Stufe (0-3) der Heizung angibt und ein `light` Element das den Status (on/off) des Lichts angibt (in dieser Reihenfolge). Optional kann ein Raum zusätzlich ein `powerplug` Element (als letztes Kindelement) enthalten.
- Ein `powerplug` Element enthält ein Beschreibung als Text und kann ein Attribut `status` mit den Werten `on`/`off` enthalten. Ist das Attribut nicht explizit gesetzt soll der Wert `on` angenommen werden.
- Wenn nicht angegeben treffen Sie plausible Annahmen über Typen von Attributen und Elementen.

File **smarthome.dtd**:

Aufgabe 4: (10)

Betrachten Sie die folgenden XPath-Abfragen angewandt auf das Dokument **smarthome.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(//user[uname])
```

```
/smarthome/user[@username=//room/@owner]/@username
```

```
/smarthome/user[3]/*[1]
```

```
/smarthome/user[1][uname]
```

```
//room[@owner=//user/@username and temperature>22]/name/text()
```

Aufgabe 5: (8)

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

```
public class RunSAX extends DefaultHandler {
    private String eleText, fName, uName, rid, first, last;
    private List<String> lr = new LinkedList<String>();
    private HashMap<String, String> n = new HashMap<String, String>();

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

    public void startElement(String namespaceURI, String localName,
                             String qName, Attributes atts) throws SAXException {
        if ("user".equals(localName))
            uName = atts.getValue("username");

        if ("room".equals(localName))
            rid = atts.getValue("id");
    }

    public void endElement(String namespaceURI, String localName,
      String qName) throws SAXException {
        if ("user".equals(localName))
            n.put(uName, fName);

        if ("firstname".equals(localName))
            first = eleText;
        if ("lastname".equals(localName))
            last = eleText;
        if ("uname".equals(localName))
            fName = last + ",␣" + first;

        if ("fullname".equals(localName))
            fName = eleText;
        if ("light".equals(localName) && eleText.equals("on"))
            lr.add(rid);
    }

    public void endDocument() throws SAXException {
        for(String u : n.keySet())
            System.out.println(u + "␣--␣" + n.get(u));

        System.out.println("Lum:␣" + lr.size());
    }

    public static void main(String[] args) throws Exception {
        InputSource source = new InputSource(new FileInputStream("smarthome.xml"));

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

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

Geben Sie hier die Ausgabe an:

Aufgabe 6: (8)

Betrachten Sie folgende XQuery **smarthome.xql**:

```
<heat>
{
  for $s in doc("smarthome.xml")/smarthome
  let $t := $s//room/temperature
  return <sh>
  <tr><l>{min($t)}</l>
      <h>{max($t)}</h>
  </tr>
  {
    for $r in $s//room
    where $r/radiator_level > 1
    order by $r/@owner descending
    return <l rn="{$r/name}">{$r/radiator_level}</l>
  }</sh>

}
</heat>
```

Geben Sie nun die Ausgabe von **smarthome.xql** angewandt auf **smarthome.xml** an.

Sie müssen sich nicht um Whitespaces kümmern.

Aufgabe 7: (10)

Betrachten Sie folgende Text Ausgabe **smarthome.xslt.out**:

```
Room:  Living room

Room:  Bedroom

Room:  Bart's room
Owner:  Bart Simpson

Room:  Lisa's room
Owner:  Lisa Simpson
Device: Reading light
```

Es handelt sich hier also um eine Zusammenfassung aller Räume in unserem smart home, wobei der "owner" ausgegeben werden soll, falls zu einem Raum einer vorhanden ist. Darüber hinaus sollen auch alle Geräte ausgegeben werden, die im jeweiligen Raum angeschlossen sind (durch "powerplug" angegeben) und auch gerade den Zustand "on" haben.

Geben Sie nun ein XSLT Dokument an, das angewandt auf **smarthome.xml**, die Ausgabe **smarthome.xslt.out** produziert. Es geht hierbei darum, das in der Angabe beschriebene Verhalten im Bezug auf den Inhalt von **smarthome.xml** zu reproduzieren, z.B. sollte ihr XSLT Dokument **nicht** statisch immer dieselbe Ausgabe produzieren, auch wenn die XML Datei grundlegend verändert werden würde.

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









</xsl:stylesheet>
```

Total points: 75

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

```
<?xml version="1.0" encoding="utf-8"?>
<smarthome>
    <address>Evergreen Terrace, Springfield</address>
    <description>
        This is the home of Marge and <ref user="Homer">Homer Simpson</ref>
        with a nice couch in the <ref room="r1">living room</ref>.
    </description>
    <user username="Marge">
        <fullname>Marge Simpson</fullname>
    </user>
    <user username="Homer">
        <uname>
            <firstname>Homer</firstname>
            <lastname>Simpson</lastname>
        </uname>
    </user>
    <room id="r1">
        <name>Living room</name>
        <temperature>21</temperature>
        <radiator_level>2</radiator_level>
        <light>on</light>
    </room>
    <room id="r2">
        <name>Bedroom</name>
        <temperature>26</temperature>
        <radiator_level>2</radiator_level>
        <light>on</light>
    </room>
    <user username="Bart">
        <fullname>Bart Simpson</fullname>
    </user>
    <room id="r3" owner="Bart">
        <name>Bart's room</name>
        <temperature>21</temperature>
        <radiator_level>1</radiator_level>
        <light>off</light>
        <powerplug status="off">NES</powerplug>
    </room>
    <user username="Lisa">
        <uname>
            <firstname>Lisa</firstname>
            <lastname>Simpson</lastname>
        </uname>
    </user>
    <room id="r4" owner="Lisa">
        <name>Lisa's room</name>
        <temperature>25</temperature>
        <radiator_level>3</radiator_level>
        <light>on</light>
        <powerplug>Reading light</powerplug>
    </room>
</smarthome>
```