Gruppe A	PRÜFUNG AUS "SEMISTRUKTURIERTE DATEN" **181.135**		15. 01. 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 Studentenausweis bereit.

Aufgabe 1: (9)

Betrachten Sie die folgende XML-Schema Datei **test.xsd**:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="z" type="rektype"/>
  <xsd:complexType name="rektype">
    <xsd:choice>
       <xsd:sequence minOccurs="2" maxOccurs="2">
          <xsd:element name="s" type="xsd:string"/>
       </xsd:sequence>
       <xsd:element name="y" type="rektype"/>
       <xsd:element name="t">
          <xsd:complexType mixed="true">
             <xsd:sequence>
                <xsd:element name="x" minOccurs="0" type="rektype"/>
                <xsd:element name="s" minOccurs="0" maxOccurs="unbounded"
                              type="xsd:string"/>
             </xsd:sequence>
          </xsd:complexType>
       </xsd:element>
    </xsd:choice>
  </xsd:complexType>
</xsd:schema>
```

Betrachten Sie weiters die unten angeführten XML-Dateien.

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. `<z><t></t></z>` gültig ⊗ ungültig ◯
2. `<z><y><s>abc</s><s>def</s></y></z>` gültig ⊗ ungültig ◯
3. `<z><t><s>abc</s><s>def</s></t></z>` gültig ⊗ ungültig ◯
4. `<z><y><y><t>abc</t><t>def</t></y></y></z>` gültig ◯ ungültig ⊗
5. `<z><t><y><t>abcdef</t></y></t></z>` gültig ◯ ungültig ⊗
6. `<z><y><y><t>abcdef</t></y></y></z>` gültig ⊗ ungültig ◯

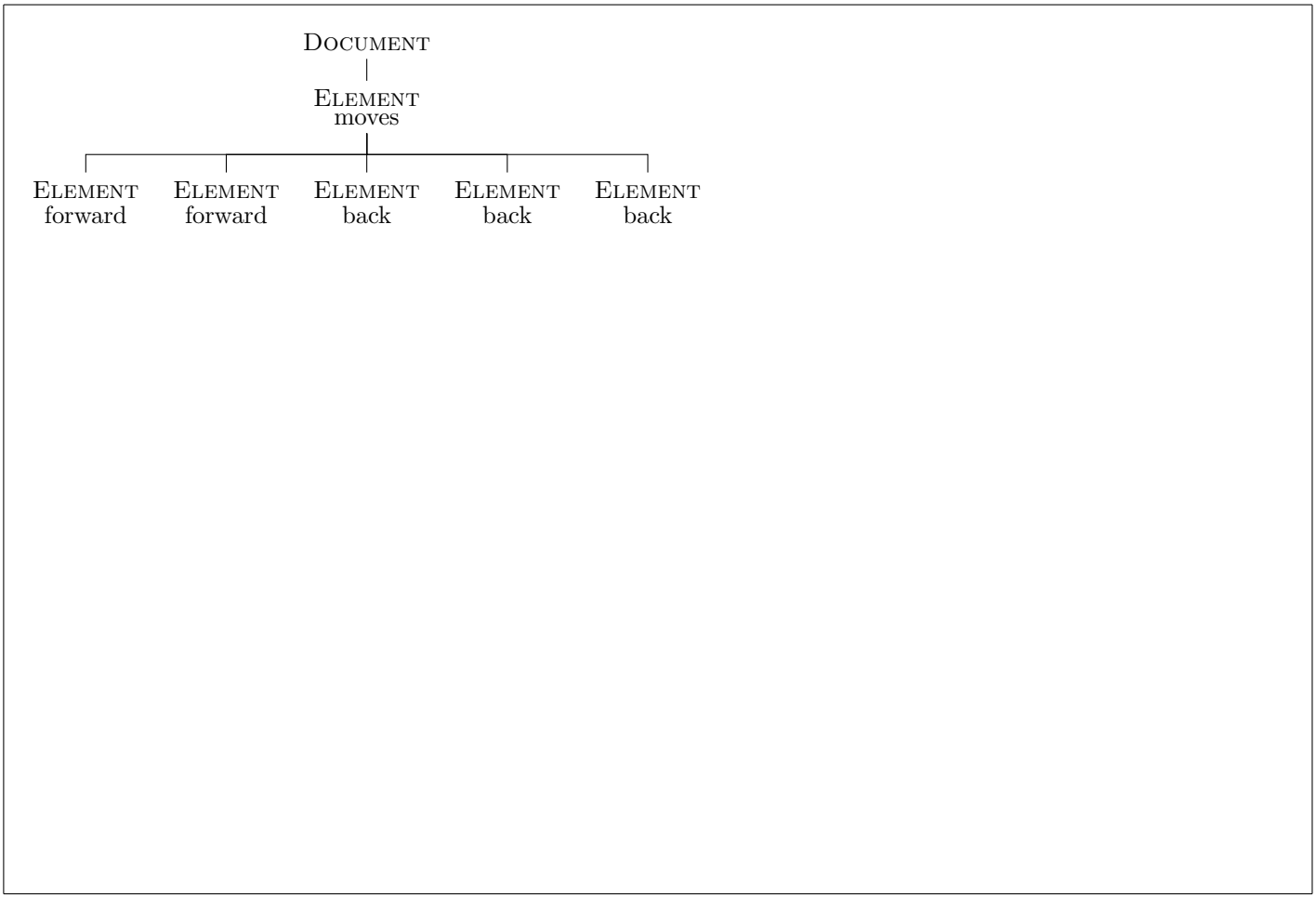
(Pro korrekter Antwort 1.5 Punkte, **pro inkorrekter Antwort -1.5 Punkte**, pro nicht beantworteter Frage 0 Punkte, für die gesamte Aufgabe mindestens 0 Punkte)

Aufgabe 2: (9)

Nehmen Sie an, dass folgende Java Method (links) mit dem Wurzelknoten des XML-Dokuments (rechts) aufgerufen wird (zB mit `secret(documentNode.getDocumentElement());`). Zeichnen Sie den resultierenden DOM-Baum *nach* der Ausführung des Codes auf.

```
private static void secret(Node root) {
  NodeList nl = root.getChildNodes();
  for(int i=0; i < nl.getLength(); i++) {
    Node t = nl.item(i);
    if(t.getNodeType() == Node.ELEMENT_NODE) {
      // lesen von Attribut n und umwandeln in Integer
      Node attr = t.getAttributes().getNamedItem("n");
      int num = Integer.parseInt(attr.getNodeValue());
      // Attribut entfernen
      t.getAttributes().removeNamedItem("n");
      for(int j=0; j < num; j++) {
        root.insertBefore(t.cloneNode(false),t);
        i++;
      }
    }
  }
}
```

```
<?xml version="1.0"?>
<moves>
  <forward n="1"/>
  <back n="2"/>
</moves>
```



Aufgabe 3: (8)

Nehmen Sie an, dass folgender `XMLFilter` auf das angegebene XML-Dokument ausgeführt wird. Schreiben Sie auf, welche SAX-Events die Applikation weitergereicht bekommt.

Whitespaces und den Präfix bei qualifizierte Namen (`qName`) können Sie ignorieren. *Namespaces*, *Text* und *Elementnamen* sollten Sie berücksichtigen. Eventnamen dürfen Sie abkürzen.

```
<?xml version="1.0" ?>
<root>
  <?exam pi?>
  <a><b/>text</a>
</root>
```

```
public class SSDFilter extends XMLFilterImpl {
  public void startDocument() throws SAXException {
    super.startDocument();
    startElement("namespace", "element", "element", null);
  }
  public void endDocument() throws SAXException {
    endElement("namespace", "element", "element");
    super.endDocument();
  }
  public void startElement(String uri, String localName, String qName,
      Attributes atts) throws SAXException {
    super.startElement("uri:001", localName+"a", qName+"a", atts);
  }
  public void endElement(String uri, String localName, String qName)
      throws SAXException {
    super.endElement("uri:001", localName+"a", qName+"a");
  }
  public void processingInstruction(String arg0, String arg1)
      throws SAXException {
  }
}
```

Eventname	Namespace	Elementname, Textinhalt etc...
`startDocument`		
`startElement`	uri:001	elementa
`startElement`	uri:001	roota
`startElement`	uri:001	aa
`startElement`	uri:001	ba
`endElement`	uri:001	ba
`text`		text
`endElement`	uri:001	aa
`endElement`	uri:001	roota
`endElement`	uri:001	elementa
`endDocument`		

Aufgabe 4: (9)

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

```
<?xml version="1.0" encoding="UTF-8"?>
<A xmlns:ns1="uri1">
  <ns1:B xmlns="uri2" xmlns:ns2="uri2">
    <D attr1="X"/>
  </ns1:B>
  <ns1:C>
    <E ns1:attr2="Y"/>
  </ns1:C>
</A>
```

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

1. Element `D` und Attirbut `attr1` liegen im selben Namespace. wahr ◯ falsch ⊗
2. Element `E` und Attirbut `attr2` liegen im selben Namespace. wahr ◯ falsch ⊗
3. Elemente `A` und `E` liegen im selben Namespace. wahr ⊗ falsch ◯
4. Element `A` liegt im leeren Namespace. wahr ⊗ falsch ◯
5. In **ns.xml** bleibt der Default Namespace unverändert. wahr ◯ falsch ⊗
6. In **ns.xml** wird ein Target-Namespace definiert. 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 5: (12)

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

1. HTML ist eine XML-Applikation. wahr ◯ falsch ⊗
2. In CDATA-Sections werden Tags üblicherweise als Text (Zeichendaten) interpretiert. wahr ⊗ falsch ◯
3. Der Speicherbedarf eines DOM-Parsers ist unabhängig von der Größe des geparsten XML-Files. wahr ◯ falsch ⊗
4. Eine XML-Schema Definition (XSD-Datei) muss ein wohlgeformtes XML-Dokument sein. wahr ⊗ falsch ◯
5. Ein XSLT-Stylesheet muss ein wohlgeformtes XML-Dokument sein. wahr ⊗ falsch ◯
6. Die DTD Elementdeklaration `(A+|B+|C+)` ist gleichwertig mit `(A|B|C)+`. wahr ◯ falsch ⊗
7. Die DTD Elementdeklaration `(A?|B?|C?)` ist gleichwertig mit `(A|B|C)?`. 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)

Aufgabe 6: (8)

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

- Ein Freundeskreis soll aus zwei oder mehr Personen bestehen.
- Jede Person hat einen *eindeutigen* Code und ein Geschlecht ("m" oder "w").
- Jedes Element `person` beinhaltet genau ein Element `name` gefolgt von beliebig vielen `status`-Elementen.
- Der Inhalt des `status`-Elementes soll gemischt sein; es dürfen als Subelemente `likes`- und `dislikes`-Elemente auftreten.
- `likes` und `dislikes` sollen keinen Inhalt haben
- Stellen Sie sicher, dass das `code`-Attribut von `likes` (bzw. von `dislikes`) nur Codes enthält, die zu einer Person passen.
- Alle Attribute sind verpflichtend und wenn nicht anders spezifiziert vom Typ `CDATA`.

Datei **freundeskreis.dtd**:

```
<?xml version="1.0" encoding="UTF-8"?>

<!ELEMENT freundeskreis (person, person+)>
<!ELEMENT person (name,status*)>
<!ATTLIST person code ID #REQUIRED>
<!ATTLIST person geschlecht (m|w) #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT status (#PCDATA|likes|dislikes)*>
<!ATTLIST status datum CDATA #REQUIRED>
<!ELEMENT likes EMPTY>
<!ELEMENT dislikes EMPTY>
<!ATTLIST likes code IDREF #REQUIRED>
<!ATTLIST dislikes code IDREF #REQUIRED>
```

Aufgabe 7: (12)

Betrachten Sie die folgenden drei XSLT-Stylesheets. Geben Sie jeweils den Output an, den das entsprechende Stylesheet angewandt auf **freundeskreis.xml** (siehe Anhang) liefert. Sie brauchen sich dabei nicht um Whitespaces etc. kümmern.

Anmerkung: Pro Teilaufgabe sind jeweils 4 Punkte erreichbar.

Nehmen Sie an dass jede Datei mit einem korrekten Header versehen ist. zB

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

Datei **query1.xsl**:

```
<xsl:output method="text" encoding="UTF-8" version="1.0" />

<xsl:template match="/">
    <xsl:apply-templates select="//person[name='Dominic']/status"/>
</xsl:template>

<xsl:template match="likes | dislikes">
    <xsl:variable name = "code" select="@code"/>
    <xsl:copy-of select="//person[@code=$code]/name"/>
</xsl:template>

</xsl:stylesheet>
```

Geben Sie hier den Output von **query1.xsl** angewandt auf **freundeskreis.xml** an:

```
... ist jetzt mit <name>Erika</name> und <name>Bruno</name> befreundet.

... hat mit <name>Hanna</name> gestritten.
```

Datei **query2.xsl**:

```
<xsl:output method="xml" indent="yes" encoding="UTF-8" version="1.0" />

<xsl:template match="/">
    <xsl:apply-templates select="//person[name='Dominic']/status"/>
</xsl:template>

<xsl:template match="likes | dislikes">
    <xsl:variable name = "code" select="@code"/>
    <xsl:apply-templates select="//person[@code=$code]/name"/>
</xsl:template>
```

Geben Sie hier den Output von **query2.xsl** angewandt auf **freundeskreis.xml** an:

```
... ist jetzt mit Erika und Bruno befreundet.

... hat mit Hanna gestritten.
```

Datei **query3.xsl**:

```
<xsl:output method="text" encoding="UTF-8" version="1.0" />

<xsl:template match="/">
    <xsl:apply-templates select="//person[name='Dominic']/status"/>
</xsl:template>

<xsl:template match="likes">
    <xsl:variable name = "code" select="@code"/>
    <xsl:copy-of select="//person[@code=$code]/name"/>
    <xsl:apply-templates select="//person[@code=$code]/status"/>
</xsl:template>

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

Geben Sie hier den Output von **query3.xsl** angewandt auf **freundeskreis.xml** an:

```
<name>Erika</name>
<name>Jutta</name>
<name>Bruno</name>
<name>Bruno</name>
```

Aufgabe 8: (8)

Schreiben Sie XPath-Anfragen für folgende Aufgabenstellungen. Zu jeder Abfrage ist ein Beispiel mit der erwarteten Ausgabe (bezgl. `freundeskreis.xml`) angegeben. Die Abfragen sollen auf allen Dokumenten, die gültig bezgl. `freundeskreis.dtd` sind, funktionieren.

1. Ausgabe der Namen aller weiblichen Personen.

```
<name>Erika</name>
<name>Hanna</name>
<name>Jutta</name>
<name>Nadia</name>
```

```
//person[@geschlecht = 'w']/name
```

2. Berechnung der Anzahl der Statusmeldungen von den Personen mit Namen Erika oder Jutta

```
3
```

```
count(//person[name='Erika']/status | //person[name='Jutta']/status)
```

3. Ausgabe der Namen der Personen, die sich selbst mögen (d.h. es existiert ein entsprechendes `likes`-Element in den Status-meldungen dieser Person).

```
<name>Karl-Heinz</name>
```

```
//person[@code=status/likes/@code])/name
```

4. Ausgabe der Namen jener Personen, die von weiblichen Personen gemocht werden.

```
<name>Jutta</name>
<name>Mario</name>
<name>Bruno</name>
```

```
//person[@code=//person[@geschlecht='w']/status/likes/@code]/name
```

Sie können diese Seite abtrennen!

Datei **freundeskreis.xml**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE freundeskreis SYSTEM "freundeskreis.dtd">
<freundeskreis>
  <person code="D" geschlecht="m">
    <name>Dominic</name>
    <status datum="20090103">
       ... ist jetzt mit <likes code="E"/> und <likes code="B"/> befreundet.
    </status>
    <status datum="20090106">
       ... hat mit <dislikes code="H"/> gestritten.
    </status>
  </person>
  <person code="E" geschlecht="w">
    <name>Erika</name>
    <status datum="20090102">
       ... will nicht mit <dislikes code="D"/> befreundet sein.
    </status>
    <status datum="20090108">
       ... mag <likes code="J"/>.
    </status>
  </person>
  <person code="H" geschlecht="w">
    <name>Hanna</name>
    <status datum="20090108">
       ... mag <likes code="M"/>.
    </status>
  </person>
  <person code="J" geschlecht="w">
    <name>Jutta</name>
    <status datum="20090102">
       ... mag <likes code="B"/>, aber mag nicht <dislikes code="KH"/>.
    </status>
  </person>
  <person code="M" geschlecht="m">
    <name>Mario</name>
    <status datum="20090108">
       ... mag <likes code="H"/>.
    </status>
  </person>
  <person code="B" geschlecht="m">
    <name>Bruno</name>
  </person>
  <person code="N" geschlecht="w">
    <name>Nadia</name>
    <status datum="20090108">
       ... ist von <dislikes code="D"/>, <dislikes code="E"/> und
       <dislikes code="H"/> nicht sehr angetan.
    </status>
  </person>
  <person code="KH" geschlecht="m">
    <name>Karl-Heinz</name>
    <status datum="20090108">
       ... ist von sich selbst <likes code="KH"/> begeistert.
    </status>
  </person>
</freundeskreis>
```

Gesamtpunkte: 75