EXAM IN "SEMI-STRUCTURED DATA" **184.705**			10. 01. 2017
Study Code	Student Id	Family Name	First Name

Working time: 100 minutes.
Exercises have to be solved on this exam sheet; Additional slips of paper will not be graded.
First, please fill in your name, study code and student number. Please, prepare your student id.

Exercise 1: (12)

Consider the following XML schema file **test.xsd**:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="W">
        <xsd:complexType mixed="false">
            <xsd:sequence>
                <xsd:element name="A" type="xsd:boolean" maxOccurs="2"/>
                <xsd:element name="B" type="xsd:int" minOccurs="0"/>
                <xsd:element name="C" type="typeC" maxOccurs="1"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:complexType name="typeC" mixed="true">
        <xsd:choice>
            <xsd:element name="D" type="xsd:boolean"/>
            <xsd:sequence>
                <xsd:element name="B" type="xsd:int" maxOccurs="2"/>
            </xsd:sequence>
        </xsd:choice>
    </xsd:complexType>
</xsd:schema>
```

Furthermore, consider the eight different XML files, which are listed below.

You may assume that each of the following XML files is well-formed. The point is to determine the validity according to **test.xsd**.

Check which of the following XML files are valid according to **test.xsd**.

1. `<W><A>true</A><B>1</B><C><D>true</D></C></W>` valid ⊗ invalid ◯
2. `<A>true</A><B>1</B><C><D>false</D></C>` valid ◯ invalid ⊗
3. `<W><A>true</A><C><B>123</B></C></W>` valid ⊗ invalid ◯
4. `<W><A/><C><B>123</B></C></W>` valid ◯ invalid ⊗
5. `<W><A>false</A><B>12</B><C><B>123</B><B>3</B></C></W>` valid ⊗ invalid ◯
6. `<W><A>false</A><B>12</B><B>2</B><C><D>true</D></C></W>` valid ◯ invalid ⊗
7. `<W><A>false</A><C><D>true</D><B>34</B></C></W>` valid ◯ invalid ⊗
8. `<W><A>false</A><C><D>true</D></C><C><D>false</D></C></W>` valid ◯ invalid ⊗

(For every correct answer 1.5 points, **for every incorrect answer -1.5 points**, for every unanswered question 0 points, you can have at least 0 points on this exercise)

Exercise 2: (15)

Decide which of the following statements are true or false.

1. Semi-structured data can be represented as labelled trees. true ⊗ false ◯
2. XML can be used as programming language and as network protocol. true ◯ false ⊗
3. DTDs are written in XML syntax. true ◯ false ⊗
4. A relational table can be represented as an XML document. true ⊗ false ◯
5. Every XQuery expression can be written as an XPath expression. true ◯ false ⊗
6. DTDs are more powerful than XML schemas. true ◯ false ⊗
7. XPath is more powerful than XSLT. true ◯ false ⊗
8. SAX is a tree-based API for manipulating XML documents. true ◯ false ⊗
9. Tree-based parsers are faster than event-based parsers. true ◯ false ⊗
10. Namespaces can be used for disambiguating elements and attributes. true ⊗ false ◯

(For every correct answer 1.5 points, **for every incorrect answer -1.5 points**, for every unanswered question 0 points, you can have at least 0 points on this exercise)

The following Exercises 3 – 7 are referring to the XML document report.xml, which can be found on the last page of this exam.

Exercise 3: (12)

Complete the DTD **report.dtd**, so that XML documents structured like **report.xml** (see attachment) are valid according to this DTD. Consider the following points when creating the DTD:

- `report` contains exactly one `content` element, at least one `author` element, and any number of `appendix` elements.
- Authors are stated before the `content` element while appendices are stated after the `content` element.
- The `content` element has mixed content, sub-elements are `authorref`, `ref`, and `section`.
- The `section` element has mixed content, sub-elements are `authorref`, and `ref`.
- The `section` element has a required attribute `title`
- `author` elements have a sub-element `name` and a required attribute `id` with a unique attribute value.
- `ref` elements don't have any content, merely a required attribute `id` which refers to the `id` of a part element.
- `authorref` elements have a required attribute `id` which refers to the `id` of a part element.
- `appendix` elements have mixed content with sub-elements `authorref`, `ref`, a required attribute `id` with a unique attribute value and a required attribute `title`.
- If not specified make reasonable assumptions on the types.

File **report.dtd**:

```
<!ELEMENT report (author+,content,appendix*)>


<!ELEMENT content (#PCDATA | authorref | ref | section)*>

<!ELEMENT section (#PCDATA | authorref | ref)*>
<!ATTLIST section title CDATA #REQUIRED>


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

<!ELEMENT authorref (#PCDATA)>
<!ATTLIST authorref id IDREF #REQUIRED>
<!ELEMENT ref EMPTY>
<!ATTLIST ref id IDREF #REQUIRED>

<!ELEMENT appendix (#PCDATA | authorref | ref)*>
<!ATTLIST appendix id ID #REQUIRED>
<!ATTLIST appendix title CDATA #REQUIRED>
```

Exercise 4: (10)

Consider the following XPath queries applied to the document **report.xml** (see attachement).

- If the given XPath expression selects the empty node set, write as output "empty output"
- If a number is selected as the result (count), write as output this number.

Now give the outputs of the respective XPath queries:

```
count(//@id)
```

9

```
//appendix[ref]/@id
```

app2

```
//section[count(authorref)>1]/@title
```

empty output

```
//author[not(@id=//authorref/@id)]/name/text()
```

Trillian

```
count(//content//authorref)
```

3

Exercise 5: (6)

Consider the following XQuery expression **xquery.xq**:

```
<summary>{
  for $s in //section
  return
     <section>
        {   if ( count($s/*) < 2 ) then <short /> else <long />   }
        <title>{$s/@title/string()}</title>
        {
           for $a in //author[@id=$s/authorref/@id]
           return
              <author>{$a/name/text()}</author>
        }
     </section>
}</summary>
```

Now give the output of **xquery.xq** applied to **report.xml**.

You do not need to consider whitespace issues.

```
<summary>
    <section>
        <long/>
        <title>part 1</title>
        <author>Arthur Dent</author>
    </section>
    <section>
        <short/>
        <title>part 2</title>
    </section>
</summary>
```

Exercise 6: (12)

Create an XSLT-Stylesheet **report.xsl**, which returns, applied to documents of the form **report.xml**, a text document being a LaTeX representation of the stored report:

- For the `content` element create a `\begin{document}` and a `\end{document}` markup. Keep the children of the content elements in the correct order between the `begin` and `end` markup.
- Now, keep all the text as it is, except:
  - Instead of the `authorref` element output content of the `name` element of the referred `author` element.
  - Instead of the `ref` element output the content of the `title` attribute of the referred `appendix` element.
- Before the contents of a section are printed, output `\section{%}`, where `%` is substituted by the value of the `title` attribute of the section.
- **Hint:** To output all contents of mixed elements you have to use: `<xsl:apply-templates select="text() | *" />`.

Consider the following output that your XSLT-Stylesheet **report.xsl** shall return applied to **report.xml**:

```
\begin{document}
   This report is co-authored by Arthur Dent and Ford Prefect and organized in two sections.

   \section{part 1}
      This section is based on the data collected by Arthur Dent and provided in A.

   \section{part 2}
      some text
\end{document}
```

Now write the XSLT-Stylesheet **report.xsl**. Control structures like `xsl:for-each`, `xsl:if`, etc. are **not** allowed. You do not need to consider whitespace issues.

File **report.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="text"/>

    <xsl:template match="report">
        <xsl:apply-templates select="content"/>
    </xsl:template>

    <xsl:template match="content">
        \begin{document}
        <xsl:apply-templates select="text()|*"/>
        \end{document}
    </xsl:template>

    <xsl:template match="section">
        \section{<xsl:value-of select="@title"/>}
        <xsl:apply-templates select="text()|*"/>
    </xsl:template>

    <xsl:template match="authorref">
        <xsl:variable name="id" select="@id"/>
        <xsl:apply-templates select="//author[@id=$id]"/>
    </xsl:template>

    <xsl:template match="author">
        <xsl:value-of select="name"/>
    </xsl:template>

    <xsl:template match="ref">
        <xsl:variable name="id" select="@id"/>
        <xsl:apply-templates select="//appendix[@id=$id]"/>
    </xsl:template>

    <xsl:template match="appendix">
        <xsl:value-of select="@title"/>
    </xsl:template>
</xsl:stylesheet>
```

Exercise 7: (8)

Complete the method `delAuthor`, which removes from a document of the form **report.xml** all `author` elements which are not referred by an `authorref` element. The method `delAuthor` has access to a variable `doc` containing the DOM representation of the XML document and to a variable `xPath`, which can be used to evaluate xPath expressions over `doc`.

For example, in the specific document **report.xml** the `author` element with the attribute `id` set to `author3` has to be removed by the method `delAuthor`. Make sure that your method also works for other documents!

You do not need to be concerned with error handling in this task.

```
private static XPath xPath = XPathFactory.newInstance().newXPath();
Document doc;

public void delAuthor () throws Exception {
    XPathExpression xpe = xPath.compile("//author[not(@id = //authorref/@id)]");
    NodeList list = (NodeList) xpe.evaluate(doc, XPathConstants.NODESET);

    for (int i = 0; i < list.getLength(); i++) {
        Node n = list.item(i);
        Node p = n.getParentNode();
        p.removeChild(n);
    }
}
```

Total points: 75

You may separate this page!

File **report.xml**:

```
<report>
  <author id="author1">
    <name>Arthur Dent</name>
  </author>
  <author id="author2">
    <name>Ford Prefect</name>
  </author>
  <author id="author3">
    <name>Trillian</name>
  </author>
  <content>
    This report is co-authored by <authorref id="author1">A. Dent</authorref>
    and <authorref id="author2">F. Prefect</authorref>
    and organized in two sections.
    <section title="part 1">
      This section is based on the data
      collected by <authorref id="author1">Dent</authorref>
      and provided in <ref id="app1"/>.
    </section>
    <section title="part 2">
      some text
    </section>
  </content>
  <appendix id="app1" title="A">
    some data
  </appendix>
  <appendix id="app2" title="B">
    even more data extending <ref id="app1"/>.
  </appendix>
</report>
```