

Guideline solution to Test Examination for the
course
XML-processing, methods, tools and theories
ITU

Spring 2005

(You are not required to repeat the questions in your answers!)

Question 1 (15 %)

Rewrite the HTML document below so that it contains only the logical information and write a CSS stylesheet that takes care of the physical layout. The output in a browser should look like the output of the HTML below.

```
<html>
<head>
<title>CSS assignment</title>
</head>
<body bgcolor="red">
<h1><font color="yellow">CSS Assignment</font></h1>
Here is a small HTML document.
Your job is to
<ol>
  <li><font color="blue">Separate the logical contents from the
physical layout</font></li>
  <li><font color="blue">Write a CSS stylesheet that takes care
of the physical layout</font></li>
  <li><font color="green">Rewrite the HTML document so that
it contains only the logical information</font></li>
</ol>
<b>NB:</b><em><font-style="italic">When linking your CSS stylesheet
```

with the rewritten HTML document, the output should look like this HTML document does when viewed in a browser.

```
</font></em>
</body>
</html>
```

Guideline solution

```
<html>
<head>
<title>CSS assignment</title>
<link href="style.css" rel="stylesheet" type="text/css">
</head>
<body>
<h1>CSS Assignment</h1>
Here is a small HTML document.
Your job is to
<ol>
  <li>Separate the logical contents from the
  physical layout</li>
  <li>Write a CSS stylesheet that takes care
  of the physical layout</li>
  <li class="last">Rewrite the HTML document so that
  it contains only the logical information</li>
</ol>
<div class="NB">NB:</div><em>When linking your CSS stylesheet
  with the rewritten HTML document, the output should look like this
  HTML document does when viewed in a browser.
</em>
</body>
</html>
```

```
style.css:
body {background-color: red}
h1 {color: yellow}
li {color: blue}
li.last {color: green}
div.NB {font-weight: bold}
em {font-style: italic}
```

Question 2 (25%)

```
<?xml version="1.0"?>
<!DOCTYPE context [
```

```

<!ELEMENT context ((building | pda)*)>
<!ELEMENT building (info?, floor*)>
<!ATTLIST building name ID #REQUIRED>
<!ELEMENT floor (info?, (room | pda)*)>
<!ATTLIST floor level ( -1 | 0 | 1 | 2 | 3 | 4 | 5) #REQUIRED>
<!ELEMENT room (info?, pda*)>
<!ATTLIST room type (canteen|office) #REQUIRED
              name CDATA #IMPLIED>
<!ELEMENT pda EMPTY>
<!ATTLIST pda name ID #REQUIRED>
<!ELEMENT info (#PCDATA)>
]>
<context>
  <building name="ITU">
    <floor level="0">
      <room name="eatIT" type="canteen">
        <info>Todays menu is hot tomato soup with bread crumbles!</info>
        <pda name="hniss"/>
        <pda name="hilde"/>
        <pda name="tofte"/>
      </room>
    </floor>
    <floor level="4">
      <info>At this floor you find offices of researchers</info>
      <room name="Hildebrandt" type="office">
        <info>Office hours every monday between 10 and 11</info>
      </room>
    </floor>
  </building>
  <pda name="guest"/>
  <building name="KUA">
    <floor level="0">
      <room name="Kantinen" type="canteen">
        <info>Today we have a buffet with chicken, salmon and veg. lasagne</info>
        <pda name="peter"/>
        <pda name="susan"/>
      </room>
    </floor>
  </building>
</context>

```

Guide-line solution:

1. the floor content must start with an optional info element followed by an arbitrary sequence of room and pda elements, so
 - (a) (info, room, pda, pda, room) is valid content

- (b) (room, info, pda) is invalid, room can not be before info
 - (c) (info, info, room) is invalid, we can not have two info elements
 - (d) () is valid (no info and no sequence)
2. It is NOT allowed to have character data as content of the `floor` element because it is not of the form "mixed content" (page 96 in the lecture notes)
 3. To allow a building having an extra, optional attribute named `type` with values `public` or `private` and default value `public` the line

```
<!ATTLIST building name ID #REQUIRED>
```

in the DTD is changed to

```
<!ATTLIST building name ID #REQUIRED
                type (public | private) "public">
```

4. Yes, the XML document would be valid if the room with attribute `name="Kantinen"` was renamed by setting `name="eatIT"`, since CDATA values need not be unique in the document.
5. An XMLSchema key expressing that two rooms in the *same* building can not have the same name, but allowing two rooms in different buildings to have the same name can be defined by inserting the a key definition in the building element as follows:

```
<element name="building">
  ....
  <unique name="roomname">
    <selector xpath="//c:room"/>
    <field xpath="@name"/>
  </unique>
</element>
```

assuming that `c` refers to the target namespace. (we use the `unique` keyword and not the `key` keyword since the `name` attribute is optional, see page 152 in lecture notes)

6. It is not possible to change the DTD such that building also can have character data as content, and still exactly the same sequences of elements because mixed content must be of the form

```
(#PCDATA | e1 | e2 | ... | en)*
```

so we cannot require that there can be atmost one info element and that it must be the first element if present.

7. XMLSchema types for the floor element and the level attribute:

```
<complexType name="floor_type">
  <sequence>
    <element ref="c:info" minOccurs="0"/>
    <choice maxOccurs="unbounded">
      <element ref="c:room"/>
      <element ref="c:pda"/>
    </choice>
  </sequence>
</complexType>

<simpleType name="level_type">
<restriction base="integer">
<minInclusive value="-1"/>
<maxInclusive value="5"/>
</restriction>
</simpleType>
```

again assuming that c refers to the target namespace, and that the elements info, room and pda are defined accordingly.

Question 3 (10%)

1. The XML-document as a tree where attributes are children of their elements can be seen in Fig. 1
2. An XPath expression that selects all pda elements inside an element room with attribute type="canteen" descendant of a building element with attribute name="ITU".

```
./building[@name="ITU"]//room[@type="canteen"]/pda
```

Question 4 (20%)

1. The output of applying the XSLT stylesheet

```
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="context">
<canteens>
<xsl:apply-templates select="//room[@type='canteen']"/>
</canteens>
```

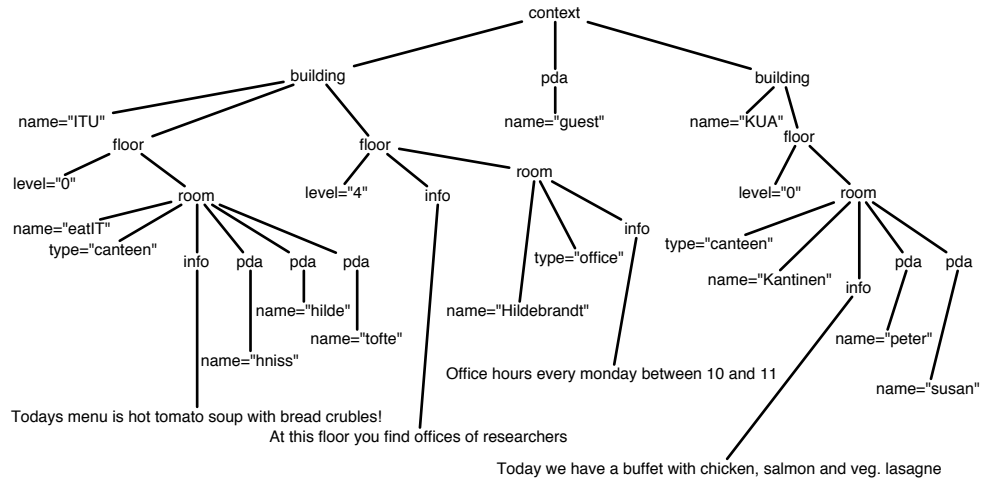


Figure 1: The XML-document as a tree where attributes are children of their elements

```

</xsl:template>

<xsl:template match="room[@type='canteen']">
<canteen name="{@name}">
  <xsl:value-of select="info/text()"/>
</canteen>
</xsl:template>
</xsl:stylesheet>

```

on the XML document above is

```

<canteen name="eatIT">
Todays menu is hot tomato soup with bread crumbles!
</canteen>
<canteen name="Kantinen">
Today we have a buffet with chicken, salmon and veg. lasagne
</canteen>
</canteens>

```

(probably add the header

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

)

2. An XQuery expression that shows the same information as the stylesheet above for any XML document valid according to the DTD above. (Assuming that the XQuery prolog has bound the xml document to the variable \$context).

```
<canteens>
{ for $c in $context//room[@type='canteen']
  return <canteen>$c/info/text()</canteen>
}
</canteens>
```

Question 5 (15%)

1. If you should implement an XML-editor, would you use the (J)DOM API or the SAX API? Justify your answer. Answer: I would use the JDOM API since it provides the entire document as a datastructure in memory, allowing random access for editing and manipulation. The SAX API provides access to the document (as events) in "document order" which would normally be insufficient for an editor.
2. What does a JAXB binding compiler produce when it is used to bind a schema?. Answer: It produces a package, containing a collection of classes and interfaces - a class and interface for each global element and type in the XMLSchema. It also provides an ObjectFactory class for generating objects corresponding to the elements.
3. Write the output of the Java code below if it is run on the XML document from Question 2

```
import java.io.*;
import java.util.*;
import org.jdom.xpath.*;
import org.jdom.*;
import org.jdom.input.*;
import org.jdom.output.*;

public class XPathContext {
    public static void main(String[] args) {
        try {
            Document d = new SAXBuilder().build(new File(args[0]));
            XPath p = XPath.newInstance("//room[pda/@name='hilde']");
            ListIterator i = p.selectNodes(d).listIterator();
            Element clean = new Element("hilde");
            while (i.hasNext()) {
                Content r = (Content)i.next();
                clean = clean.addContent((Content)r.clone());
            }
            Document n = new Document(clean);
            new XMLOutputter().output(n, System.out);
        } catch (Exception e) {e.printStackTrace();}
    }
}
```

Output:

```
<hilde>
<room name="eatIT" type="canteen">
    <info>Todays menu is hot tomato soup with bread crumbles!</info>
```

```
<pda name="hniss"/>
<pda name="hilde"/>
<pda name="tofte"/>
</room>
</hilde>
```

Question 6 (15%)

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.itu.dk/ixmr/tips" xmlns:tips="http://www.itu.dk/ixmr/tips" >

  <element name="results" type="tips:daresultstype"></element>
  <attribute name="thedata" type="date"></attribute>
  <element name="result" type="tips:result"></element>

  <simpleType name="result">
    <restriction base="string">
      <enumeration value="1"></enumeration>
      <enumeration value="X"></enumeration>
      <enumeration value="2"></enumeration>
    </restriction>
  </simpleType>

  <complexType name="resultstype">
    <sequence>
      <element maxOccurs="13" minOccurs="13" ref="tips:result"></element>
    </sequence>
  </complexType>

  <complexType name="daresultstype">
    <complexContent>
      <extension base="tips:resultstype">
        <attribute name="thedata" type="date" use="required"></attribute>
      </extension>
    </complexContent>
  </complexType>
</schema>
```

1. an xml-document with root element `results` that is valid according to the XMLSchema above

```
<?xml version="1.0" encoding="UTF-8"?>
<results thedate="2005-04-05" xmlns="http://www.itu.dk/ixmr/tips"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.itu.dk/ixmr/tips tips.xsd">
  <result>1</result>
  <result>1</result>
  <result>1</result>
  <result>1</result>
  <result>1</result>
  <result>1</result>
  <result>1</result>
```

```
<result>1</result>
<result>1</result>
<result>1</result>
<result>X</result>
<result>X</result>
<result>2</result>
<result>1</result>
</results>
```

2. Write a DTD corresponding to the XMLSchema, with root element `results`

```
<!ELEMENT results (result,result,result,result,result,
                  result,result,result,result,result,result,result)>
<!ATTLIST results thedate CDATA #REQUIRED>
<!ELEMENT result (#PCDATA)>
```

it is not possible to specify that the value of `thedate` must be a data, neither that the text inside the `result` must be 1, X or 2.