

COMP60362: SSD & XML Coursework

All written coursework is to be handed in to the postgraduate office by the following Monday, 11:30 am. In addition, please email XML document, DTDs, queries, xpaths, stylesheets, schemata, etc. to me at sattler@cs.man.ac.uk.

Files emailed are:

1. digitalmedia.xml
2. digitalmedia.dtd
3. xpaths.txt
4. digitalmedia_namespaces.xml
5. digitalmedia_namespaces.dtd

1. [5 marks] Describe in 5 sentences what the W3C is, and what "W3C recommendation" means.

- W3C (World Wide Web Consortium) is the principal 'standards organization' covering the World Wide Web technologies.
- Both full-time staff (based at MIT in USA, INRIA in France, Keio in Japan, etc), individual experts and organizations are members of the consortium.
- W3C recommendations are *finalized* versions of the internationally recognized standards for the technologies that form the basis for the World Wide Web, covering such topics as languages (like HTML), protocols (such as http://), and resource locators (like URLs).
- W3C recommendations go through a series of drafting stages before issue; at early stages anyone can comment on the recommendations, and a working group drawn from the consortium gradually defines an increasingly concrete standard.
- It is intended, but *not enforced*, that manufacturers (and others) implement these standards, free of charge, for hardware and for applications using the World Wide Web.

2. [10 marks] Start the <Oxygen/> tool installed in the MSc lab, familiarize yourself with it, choose some application domain and write a meaningful and well-formed data-centric XML document about it that contains at least 50 elements, with

- a. some elements being at least at level 5 (the document node is at level 0),
- b. at least 8 different elements, and
- c. attributes of different types including strings, ID, and IDREF.

Application domain chosen: a digital media library
'digitalmedia.xml' file is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  K. John Smith
  Exercise 2
-->

<!-- The following DTD of the types internal and external
      added for Exercise 4-->
<!DOCTYPE digitalmedia
  SYSTEM "digitalmedia.dtd"
  [
    <!ENTITY server "http://macHome9/media">
    <!ENTITY copyright "&#xA9;">
    <!ENTITY IraGlass
      '
    <author>
      <name>
        <family>Glass</family>
        <given>Ira</given>
```

```

    </name>
    <info>http://en.wikipedia.org/wiki/Ira_Glass</info>
  </author>
'>
]>
<digitalmedia>
  <spokenword>

    <book genre="drama" id="book1">
      <title>The Sea Wolf</title> {
      <publication_detail>
        <author>
          <name>
            2 <family>London</family> 2
            3 <given>Jack</given> 3
          </name>
          4 <info>http://www.jacklondon.com/</info> 4
          5 <picture>http://www.jacklondon.com/pic.jpg</picture> 5
        </author>
        6 <edition/>
        7 <publisher/>
      </publication_detail>
      <format type="audible">
        8 <location>&server;/audible/seawolf/</location>
        <source>
          9 <owner>&copyright;Audible.com</owner>
          10 <location type="incomplete">Audible.com</location>
          11 <cost>$7.49</cost>
        </source>
      </format>
      <actor>
        <name>
          12 <family>Hagon</family>
          13 <given>Garrick</given>
        </name>
      </actor>
      14 <info>http://en.wikipedia.org/wiki/The_Sea_Wolf</info>
    </book>

    <book genre="drama" id="book2">
      <title>Bartleby the Scrivener, and other stories <subtitle>Bartleby the Scrivener</subtitle>
      16 <subtitle>The Lightning-Rod Man</subtitle>
      17 <subtitle>The Bell-Tower</subtitle>
      </title>
      <publication_detail>
        <author>
          <name>
            18 <family>Melville</family>
            19 <given>Herman</given>
          </name>
          20 <info>http://en.wikipedia.org/wiki/Herman_Melville</info>
        </author>
        21 <edition/>
        22 <publisher/>
      </publication_detail>
      <format type="audible">
        23 <location>&server;/audible/bartleby/</location>
        <source>
          23 <owner>&copyright;Audible.co.uk</owner>
          24 <location type="incomplete">Audible.co.uk</location>
          25 <cost>£9.99</cost>
        </source>
      </format>
      <actor>
        <name>
          26 <family>Roberts</family>
          27 <given>William</given>
        </name>

```

more Oaws = "subords"

```

</actor>
<info>http://en.wikipedia.org/wiki/Bartleby_the_Scrivener</info>
</book>

<play genre="drama" id="play1">
  <title>An Inspector Calls</title>
  <author>
    <name>
      <family>Priestley</family>
      <given>J.B.</given>
    </name>
    <info>http://en.wikipedia.org/wiki/J._B._Priestley</info>
  </author>
  <format type="mp3">
    <location>&server;/radio/inspectorcalls.mp3</location>
    <source>
      <owner>&copyright;bbc</owner>
      <location type="incomplete">bbc7</location>
      <cost>0</cost>
    </source>
  </format>
  <actor>
    <name>
      <family>Peck</family>
      <given>Bob</given>
    </name>
    <info>http://www.imdb.com/name/nm0669629/</info>
  </actor>
  <info>http://en.wikipedia.org/wiki/An_Inspector_Calls</info>
</play>

<play genre="drama" id="play2">
  <title>'night, Mother</title>
  <author>
    <name>
      <family>Norman</family>
      <given>Marsha</given>
    </name>
  </author>
  <format type="mp3">
    <location>&server;/radio/nightmother.mp3</location>
    <source>
      <owner>&copyright;bbc</owner>
      <location type="incomplete">bbc7</location>
      <cost>0</cost>
    </source>
  </format>
  <actor>
    <name>
      <family>Gless</family>
      <given>Sharon</given>
    </name>
    <info>http://en.wikipedia.org/wiki/Sharon_Gless</info>
  </actor>
  <actor>
    <name>
      <family>Helmond</family>
      <given>Katherine</given>
    </name>
    <info>http://en.wikipedia.org/wiki/Katherine_Helmond</info>
    <info>http://www.imdb.com/name/nm0001340/</info>
  </actor>
  <info>Pulitzer Prize 1983</info>
  <info>http://en.wikipedia.org/wiki/Marsha_Norman</info>
  <info>http://en.wikipedia.org/wiki/%27night%2C_Mother</info>
</play>

<play genre="drama" id="play3">
  <title>Breaker Morant</title>

```

```

<author>
  <name>
    <given>Kenneth</given>
    <family>Ross</family>
  </name>
</author>
<format type="ra">
  <location>&server;/radio/breaker.mp3</location>
  <source>
    <owner>&copyright;bbc</owner>
    <location type="incomplete">bbc radio4</location>
    <cost>0</cost>
  </source>
</format>
<info>http://en.wikipedia.org/wiki/Breaker_Morant</info>
<link related="documentary1"/>
<link related="documentary2"/>
</play>

<documentary id="documentary1">
  <title>Trail of Tears <subtitle>TAL Episode 107</subtitle>
  </title> &IraGlass; <author>
    <name>
      <family>Vowell</family>
      <given>Sarah</given>
    </name>
  </author>
  <date month="07" day="03" year="1998"/>
  <format type="mp3">
    <location>&server;/TAL/107.mp3</location>
    <source>
      <owner>&copyright;PBS</owner>
      <location type="incomplete">http://www.thislife.org/</location>
      <cost>0</cost>
    </source>
  </format>
  <format type="FlashPlayer9">
    <location type="incomplete">http://www.thislife.org/</location>
  </format>
  <info>Cherokee journey to Oklahoma</info>
  <info>http://www.thislife.org/</info>
  <link related="documentary2"/>
  <link related="play3"/>
</documentary>

<documentary id="documentary2">
  <title>Before It Had a Name <subtitle>TAL Episode 197</subtitle>
  </title> &IraGlass; <date month="10" day="26" year="2001"/>
  <format type="mp3">
    <location>&server;/TAL/197.mp3</location>
    <source>
      <owner>&copyright;PBS</owner>
      <location type="incomplete">http://www.thislife.org/</location>
      <cost>0</cost>
    </source>
  </format>
  <format type="FlashPlayer9">
    <location type="incomplete">http://www.thislife.org/</location>
  </format>
  <info>Halocaust Interviews, and other stories</info>
  <info title="This American Life">http://www.thislife.org/</info>
  <info title="Voices of the Holocaust">http://voices.iit.edu/</info>
  <link related="documentary1"/>
  <link related="play3"/>
</documentary>

</spokenword>

<music/>

```

```
<film/>
</digitalmedia>
```

3. [7 marks] In 4 sentences, describe the role played by entities in DTDs and the special role of parameter entities. Give an example of three DTD statements involving entities and parameter entities.

- In DTDs, entities consist of variables (known as 'entity names') used to define *shortcuts/abbreviations* pointing to either predefined values (e.g. < for <), to internal 'entity values' or to external URIs.
- Entities can be referred to in the DTD in areas that will appear in the XML file, and can also be referred to directly in the corresponding XML files.
- Entities are restricted to unparsed objects, or to well-formed text without cyclic references.
- Parameter entities are entities that are local to the DTD (i.e. can only be unfolded inside the DTD), are not restricted to areas of the DTD that will appear directly in the XML file, and can be redefined/over-written in an internal DTD for each XML file.

An example of a DTD fragment involving entities is:

```
<!--This is an internal parameter entity declaration-->
<!ENTITY % one2twelve "01|02|03|04|05|06|07|08|09|10|11|12">
<!--This is an external general entity declaration.
      It can now be referred to in the XML file, for example...
      <footer>&myMail; </footer>
-->
<!ENTITY myMail SYSTEM "/commonstuff/emailaddresses.xml">
<!--usage -->
<!ATTLIST footer month (%zero2twelve;) #REQUIRED>
```

See exercises 2 and 4 for more examples.

4. [10 marks] Use <oXygen/> to design a DTD for the XML document you designed in Exercise 2. This DTD should contain

- a. at least three element declarations whose content model refers to other elements,
- b. some element declarations that make use of at least the occurrence indicators *, +, |, and ?,
- c. attribute declarations of type ID, IDREF,
- d. entities.

Explain how you designed it and use <oXygen/> to check whether the document you wrote for Exercise 2 is valid.

The 'digitalmedia.dtd' file is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  K. John Smith
  Exercise 4.
-->
<!ELEMENT digitalmedia (spokenword,music,film)>
<!ELEMENT spokenword (documentary|play|book)*>

<!ELEMENT book (title,publication_detail*,format+,actor*,info*,link*)>
<!ATTLIST book id ID #REQUIRED>
<!ATTLIST book genre (drama|misc) 'misc'>

<!ELEMENT documentary (title,author*,date?,format+,actor*,info*,link*)>
<!ATTLIST documentary id ID #REQUIRED genre (drama|misc) 'misc'>

<!ELEMENT play (title,author*,format+,actor*,info*,link*)>
<!ATTLIST play id ID #REQUIRED>
```

```

<!ATTLIST play genre (drama|misc) 'misc'>

<!ELEMENT publication_detail (author+,edition,publisher)>
<!ELEMENT edition (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>

<!ELEMENT title (#PCDATA|subtitle)*>
<!ELEMENT subtitle (#PCDATA)>
<!ELEMENT author (name,info*,picture*)>
<!ELEMENT format (location+,source*)>
<!ATTLIST format type (mp3|flv|audible|FlashPlayer9|ra) #REQUIRED>
<!ELEMENT actor (name,info*,picture*)>
<!ELEMENT info (#PCDATA)>
<!ATTLIST info title CDATA #IMPLIED>
<!ELEMENT link EMPTY>
<!ATTLIST link related IDREF #IMPLIED>

<!ELEMENT name ((family,given)|(given,family))>
<!ELEMENT family (#PCDATA)>
<!ELEMENT given (#PCDATA)>
<!ELEMENT picture (#PCDATA)>

<!ELEMENT source (owner,location?,date?,cost)>
<!ELEMENT location (#PCDATA)>
<!ATTLIST location type (incomplete|complete) #IMPLIED>
<!ELEMENT owner (#PCDATA)>
<!ELEMENT date EMPTY>
<!ENTITY % zero2twelve "01|02|03|04|05|06|07|08|09|10|11|12">
<!ATTLIST date month (%zero2twelve;) #REQUIRED>
<!ATTLIST date day (%zero2twelve;|13|14|15|16|17|18|19|20|21|22|23|24|25|26|27|28|29|30|31) #REQUIRED>
<!ATTLIST date year CDATA #REQUIRED>
<!ELEMENT cost (#PCDATA)>

<!--would hold data in a real xml file,
for now pretend it's in development-->
<!ELEMENT music (country,rap,pop)*>
<!ELEMENT film (comedy,drama)*>
<!ELEMENT country ANY>
<!ELEMENT rap ANY>
<!ELEMENT pop ANY>
<!ELEMENT comedy ANY>
<!ELEMENT drama ANY>

```

Note some of the DTD components are included in the 'digitalmedia.xml' file above as an internal DTD.

The design mechanism was as follows:

1. Create an <!ELEMENT ...> entry for each tag in the document.
2. Create an <!ATTLIST ...> entry for each tag with an attribute in the document.
3. Scan through the document to determine the most restrictive usage of elements and of attributes that will fit the .xml file (some tools automate this step).
4. Set the restrictions in the DTD file, making them as restrictive as possible, but taking account possible future extensions of the .xml file (as directed by the *meaning* of the elements/attributes).
5. Look for repeated elements that can be converted to <!ENTITY ...> statements.

- The DTD is referred to in the XML document – see exercise 2 for syntax. In this way the parser knows which DTD to use when checking the XML document. Some of the DTD is included as an internal DTD, and this is automatically taken part of the checking process (with higher priority than the external DTD).

-In <oXygen/> parser the validity of the XML document (with respect to the DTD) is confirmed by pressing the 'Reset cache and Validate' button. Messages inform the user of any problems. There is also some support for real-time feedback of validity as changes are made to the XML file.

5. [16 marks] For each of the following regular expressions E_i , determine whether each of the following statements is true or false and explain each of your answers in one or two sentences.

$$E_1 = ((a \cdot b \cdot c) + (((a \cdot b) + (a \cdot c)) \cdot c))^*$$

$$E_2 = ((a \cdot b \cdot (d + e)) + (a \cdot (b + c) \cdot f) + (a \cdot c \cdot (d + e)))^*$$

$$E_1' = [(a_1 \cdot b_1 \cdot c_1) + (((a_2 \cdot b_2) + (a_3 \cdot c_2)) \cdot c_3)]^* \text{ or } [(a_1 b_1 c_1) \{((a_2 b_2) | (a_3 c_2)) c_3\}]^*$$

$$E_2' = [(a_1 \cdot b_1 \cdot (d_1 + e_1)) + (a_2 \cdot (b_2 + c_1) \cdot f_1) + (a_3 \cdot c_2 \cdot (d_2 + e_2))]^* \\ \text{ or } [(a_1 b_1 (d_1 | e_1)) \mid (a_2 (b_2 | c_1) f_1) \mid (a_3 c_2 (d_2 | e_2))]^*$$

a. E_i is unambiguous

- (i) Consider expression, $F = abc \in L(E_1)$.
 $F = a_1 b_1 c_1$ and $F = a_2 b_2 c_3$ are possible witnesses.
 So, E_1 is unambiguous, is **false** (illustration with a counter-example).
- (ii) There are no expressions in $L(E_2)$ for which there are multiple witnesses.
 So, E_2 is unambiguous, is **true**.

b. $L(E_i)$ is unambiguous

- (i) Consider expression, $F = abc \in L(E_1)$.
 $F = a_1 b_1 c_1$ and $F = a_2 b_2 c_3$ are possible witnesses.
 So, $L(E_1)$ is unambiguous, is **false** (illustration with a counter-example).
- (ii) There are no expressions in $L(E_2)$ for which there are multiple witnesses.
 So, $L(E_2)$ is unambiguous, is **true**.

c. E_i is 1-unambiguous

- (i) E_1 is unambiguous is false, so, E_1 is 1-unambiguous is also **false** (1-ambiguity is more restrictive)
- (ii) Consider expression, $F = abdabd \in L(E_2)$.
 $F = a_1 b_1 d_1 a_1 \dots$ or $a_1 b_1 d_1 a_2 \dots$ or $a_1 b_1 d_1 a_3 \dots$ are possible pathways.
 So, E_2 is 1-unambiguous, is **false** (illustration with a counter-example).

d. $L(E_i)$ is 1-unambiguous

- (i) $L(E_1)$ is unambiguous is false, so, $L(E_1)$ is 1-unambiguous is also **false** (1-ambiguity is more restrictive)
- (ii) Consider expression, $F = abdabd \in L(E_2)$.
 $F = a_1 b_1 d_1 a_1 \dots$ or $a_1 b_1 d_1 a_2 \dots$ or $a_1 b_1 d_1 a_3 \dots$ are possible pathways.
 So, $L(E_2)$ is 1-unambiguous, is **false** (illustration with a counter-example).

6. [10 marks] Use <oXygen/> to evaluate XPath expressions on the XML document you designed in Exercise 3.

Devise two XPath location paths, each

a. consisting of at least 4 location steps

b. making use of at least two directions in a document tree (up, down and horizontally) and

c. containing at least 2 location steps with predicates, some on attributes and some on elements.

Write each path both in the original and in the abbreviated syntax. For each path, provide the nodes it selects and explain why this set is selected.

- The Xpath expression reads ...

For entries where the copyright is owned by the BBC AND the file format is mp3, select the title, AND the author's given- and family-names.

```
/descendant-or-self::node()/child::source[child::owner="&#xA9;bbc"]/parent::node()/parent::node()/child::format[attribute::type="mp3"]/parent::node()(child::title,child::author/child::name/(child::family,child::given))
```

```
//source[owner="&#xA9;bbc"]/../../format[@type="mp3"]/..(title,author/name/(family,given))
```

Nodes selected are:

```
<title>An Inspector Calls</title>
<family>Priestley</family>
<given>J.B.</given>
<title>'night, Mother</title>
<family>Norman</family>
<given>Marsha</given>
```

The selection is as follows...

```
--source-owner = @BBC gives play1(An Inspector Calls), play2('night, Mother) and
play3(Breaker Morant)
--filtering out play3 because it has format type 'ra', gives play1 and play2
--Navigating up gives the titles An Inspector Calls and 'night, Mother (selected nodes).
--Navigating to the author tag and extracting the names gives Priestley, J.B. and Norman,
Marsha (selected nodes).
```

- The Xpath expression reads ...

For entries where Ira Glass is the author, and there exists an author whose name is not Ira Glass (i.e. Ira Glass is a co-author), select the title, AND the titles of any entries related to those entries by a link.

```
/descendant-or-self::node()/child::author/child::name[child::family="Glass"][child::given="Ira"]/parent::node()/parent::node()/child::author/child::name[child::family!="Glass"][child::given!="Ira"]/parent::node()/parent::node()(child::title,(id(child::link/attribute::related))/child::title)
```

```
//author/name[family="Glass"][given="Ira"]/../../author/name[family!="Glass"][given!="Ira"]/../../(title,(id(link/@related))/title)
```

Nodes selected are:

```
<title>Breaker Morant</title>
<title>Trail of Tears <subtitle>TAL Episode 107</subtitle></title>
<title>Before It Had a Name <subtitle>TAL Episode197</subtitle></title>
```

The selection is as follows...

--Ira Glass is the author of documentary1 (Trail of Tears) and of documentary2 (Before It Had a Name)

--Ira Glass is the only author listed on documentary2, and so this is discarded. Sarah Vowel is a co-author on documentary1, which remains.

--The title of documentary1 is selected, Trail of Tears... (**selected node**).

--documentary 1 has links to documentary2 and play3, which are navigated to based on the unique ID.

--Their titles are then selected, play3 is Breaker Morant, and the previously discarded documentary2 is Before It Had a Name... (**selected nodes**).

--The order in which the nodes are reported reflects the absolute order in the XML file, play3, documentary1, documentary2.

7. [8 marks] Take the XML document from Exercise 2 and add namespace information to it so that all elements and attributes in the XML document are prefixed (either explicitly or by default). Use at least 2 different namespaces and explain your choice.

-The default namespace for this document was set to "http://smithk.cs.man.ac.uk"

-Different namespaces were used to disambiguate the 'info' element between plain-text and remote web-based documents using ...

```
xmlns:txt="http://smithk.cs.man.ac.uk/simple_text"
```

```
xmlns:web="http://smithk.cs.man.ac.uk/http"
```

-No *meaning* can be assigned to the http:// address, since it doesn't exist, but it will be unique in the world, and so this document could be used with any other and be guaranteed to have a unique namespace for all elements.

-No namespaces were assigned to attributes – there was no need to disambiguate these – and W3C recommends that this approach be adopted wherever possible.

-Changes were made to the DTD in order to accommodate the new 'xmns' attributes for the root element. Likewise for the namespace qualified entity names - handled simply treating them as new entities that just happen to have a colon in the entity name.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  K. John Smith
  Exercise 7
-->

<!-- The following DTD of the types internal and external
      added for Exercise 4-->
<!DOCTYPE digitalmedia
  SYSTEM "digitalmedia_namespaces.dtd"
  [
    <!ENTITY server "http://macHome9/media">
    <!ENTITY copyright "&#xA9;">
    <!ENTITY IraGlass
  '
    <author>
      <name>
        <family>Glass</family>
        <given>Ira</given>
      </name>
      <web:info>http://en.wikipedia.org/wiki/Ira_Glass</web:info>
    </author>
  '>
  ]>
```

```

<digitalmedia xmlns="http://smithk.cs.man.ac.uk" xmlns:txt="http://smithk.cs.man.ac.uk/simple_text"
  xmlns:web="http://smithk.cs.man.ac.uk/http">
  <spokenword>

  <book genre="drama" id="book1">
    <title>The Sea Wolf</title>
    <publication_detail>
      <author>
        <name>
          <family>London</family>
          <given>Jack</given>
        </name>
        <web:info>http://www.jacklondon.com/</web:info>
        <picture>http://www.jacklondon.com/pic.jpg</picture>
      </author>
      <edition/>
      <publisher/>
    </publication_detail>
    <format type="audible">
      <location>&server;/audible/seawolf/</location>
      <source>
        <owner>&copyright;Audible.com</owner>
        <location type="incomplete">Audible.com</location>
        <cost>$7.49</cost>
      </source>
    </format>
    <actor>
      <name>
        <family>Hagon</family>
        <given>Garrick</given>
      </name>
    </actor>
    <web:info>http://en.wikipedia.org/wiki/The_Sea_Wolf</web:info>
  </book>

  <book genre="drama" id="book2">
    <title>Bartleby the Scrivener, and other stories <subtitle>Bartleby the Scrivener</subtitle>
      <subtitle>The Lightning-Rod Man</subtitle>
      <subtitle>The Bell-Tower</subtitle>
    </title>
    <publication_detail>
      <author>
        <name>
          <family>Melville</family>
          <given>Herman</given>
        </name>
        <web:info>http://en.wikipedia.org/wiki/Herman_Melville</web:info>
      </author>
      <edition/>
      <publisher/>
    </publication_detail>
    <format type="audible">
      <location>&server;/audible/bartleby/</location>
      <source>
        <owner>&copyright;Audible.co.uk</owner>
        <location type="incomplete">Audible.co.uk</location>
        <cost>£9.99</cost>
      </source>
    </format>
    <actor>
      <name>
        <family>Roberts</family>
        <given>William</given>
      </name>
    </actor>
    <web:info>http://en.wikipedia.org/wiki/Bartleby_the_Scrivener</web:info>
  </book>

  <play genre="drama" id="play1">

```

```

<title>An Inspector Calls</title>
<author>
  <name>
    <family>Priestley</family>
    <given>J.B.</given>
  </name>
  <web:info>http://en.wikipedia.org/wiki/J.\_B.\_Priestley</web:info>
</author>
<format type="mp3">
  <location>&server;/radio/inspectorcalls.mp3</location>
  <source>
    <owner>&copyright;bbc</owner>
    <location type="incomplete">bbc7</location>
    <cost>0</cost>
  </source>
</format>
<actor>
  <name>
    <family>Peck</family>
    <given>Bob</given>
  </name>
  <web:info>http://www.imdb.com/name/nm0669629/</web:info>
</actor>
<web:info>http://en.wikipedia.org/wiki/An\_Inspector\_Calls</web:info>
</play>

<play genre="drama" id="play2">
<title>'night, Mother</title>
<author>
  <name>
    <family>Norman</family>
    <given>Marsha</given>
  </name>
</author>
<format type="mp3">
  <location>&server;/radio/nightmother.mp3</location>
  <source>
    <owner>&copyright;bbc</owner>
    <location type="incomplete">bbc7</location>
    <cost>0</cost>
  </source>
</format>
<actor>
  <name>
    <family>Gless</family>
    <given>Sharon</given>
  </name>
  <web:info>http://en.wikipedia.org/wiki/Sharon\_Gless</web:info>
</actor>
<actor>
  <name>
    <family>Helmond</family>
    <given>Katherine</given>
  </name>
  <web:info>http://en.wikipedia.org/wiki/Katherine\_Helmond</web:info>
  <web:info>http://www.imdb.com/name/nm0001340/</web:info>
</actor>
<txt:info>Pulitzer Prize 1983</txt:info>
<web:info>http://en.wikipedia.org/wiki/Marsha\_Norman</web:info>
<web:info>http://en.wikipedia.org/wiki/%27night%2C\_Mother</web:info>
</play>

<play genre="drama" id="play3">
<title>Breaker Morant</title>
<author>
  <name>
    <given>Kenneth</given>
    <family>Ross</family>
  </name>

```

```

</author>
<format type="ra">
  <location>&server;/radio/breaker.mp3</location>
  <source>
    <owner>&copyright;bbc</owner>
    <location type="incomplete">bbc radio4</location>
    <cost>0</cost>
  </source>
</format>
<web:info>http://en.wikipedia.org/wiki/Breaker_Morant</web:info>
<link related="documentary1"/>
<link related="documentary2"/>
</play>

<documentary id="documentary1">
  <title>Trail of Tears <subtitle>TAL Episode 107</subtitle>
  </title> &IraGlass; <author>
    <name>
      <family>Vowell</family>
      <given>Sarah</given>
    </name>
  </author>
  <date month="07" day="03" year="1998"/>
  <format type="mp3">
    <location>&server;/TAL/107.mp3</location>
    <source>
      <owner>&copyright;PBS</owner>
      <location type="incomplete">http://www.thislife.org/</location>
      <cost>0</cost>
    </source>
  </format>
  <format type="FlashPlayer9">
    <location type="incomplete">http://www.thislife.org/</location>
  </format>
  <txt:info>Cherokee journey to Oklahoma</txt:info>
  <web:info>http://www.thislife.org/</web:info>
  <link related="documentary2"/>
  <link related="play3"/>
</documentary>

<documentary id="documentary2">
  <title>Before It Had a Name <subtitle>TAL Episode 197</subtitle>
  </title> &IraGlass; <date month="10" day="26" year="2001"/>
  <format type="mp3">
    <location>&server;/TAL/197.mp3</location>
    <source>
      <owner>&copyright;PBS</owner>
      <location type="incomplete">http://www.thislife.org/</location>
      <cost>0</cost>
    </source>
  </format>
  <format type="FlashPlayer9">
    <location type="incomplete">http://www.thislife.org/</location>
  </format>
  <txt:info>Halocaust Interviews, and other stories</txt:info>
  <web:info title="This American Life">http://www.thislife.org/</web:info>
  <web:info title="Voices of the Holocaust">http://voices.iit.edu/</web:info>
  <link related="documentary1"/>
  <link related="play3"/>
</documentary>

</spokenword>

<music/>
<film/>
</digitalmedia>

```

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

```

<!--
  K. John Smith
  Exercise 7.
-->
<!ELEMENT digitalmedia (spokenword,music,film)>
<!ELEMENT spokenword (documentary|play|book)*>

<!-- changes for namespaces -->
<!ATTLIST digitalmedia xmlns CDATA #IMPLIED>
<!ATTLIST digitalmedia xmlns:txt CDATA #IMPLIED>
<!ATTLIST digitalmedia xmlns:web CDATA #IMPLIED>
<!ELEMENT documentary (title,author*,date?,format+,actor*,txt:info*,web:info*,link*)>
<!ELEMENT book (title,publication_detail*,format+,actor*,txt:info*,web:info*,link*)>
<!ELEMENT author (name,txt:info*,web:info*,picture*)>
<!ELEMENT web:info (#PCDATA)>
<!ATTLIST web:info title CDATA #IMPLIED>
<!ELEMENT txt:info (#PCDATA)>
<!ATTLIST txt:info title CDATA #IMPLIED>
<!ELEMENT actor (name,txt:info*,web:info*,picture*)>
<!ELEMENT play (title,author*,format+,actor*,txt:info*,web:info*,link*)>
<!-- changes for namespaces -->

<!--<!ELEMENT book (title,publication_detail*,format+,actor*,info*,link*)>-->
<!ATTLIST book id ID #REQUIRED>
<!ATTLIST book genre (drama|misc) 'misc'>

<!--<!ELEMENT documentary (title,author*,date?,format+,actor*,info*,link*)>-->
<!ATTLIST documentary id ID #REQUIRED genre (drama|misc) 'misc'>

<!--<!ELEMENT play (title,author*,format+,actor*,info*,link*)>-->
<!ATTLIST play id ID #REQUIRED>
<!ATTLIST play genre (drama|misc) 'misc'>

<!ELEMENT publication_detail (author+,edition,publisher)>
<!ELEMENT edition (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>

<!ELEMENT title (#PCDATA|subtitle)*>
<!ELEMENT subtitle (#PCDATA)>
<!--<!ELEMENT author (name,info*,picture*)>-->
<!ELEMENT format (location+,source*)>
<!ATTLIST format type (mp3|flv|audible|FlashPlayer9|ra) #REQUIRED>
<!--<!ELEMENT actor (name,info*,picture*)>-->
<!--<!ELEMENT info (#PCDATA)>-->
<!--<!ATTLIST info title CDATA #IMPLIED>-->
<!ELEMENT link EMPTY>
<!ATTLIST link related IDREF #IMPLIED>

<!ELEMENT name ((family,given)|(given,family))>
<!ELEMENT family (#PCDATA)>
<!ELEMENT given (#PCDATA)>
<!ELEMENT picture (#PCDATA)>

<!ELEMENT source (owner,location?,date?,cost)>
<!ELEMENT location (#PCDATA)>
<!ATTLIST location type (incomplete|complete) #IMPLIED>
<!ELEMENT owner (#PCDATA)>
<!ELEMENT date EMPTY>
<!ENTITY % zero2twelve "01|02|03|04|05|06|07|08|09|10|11|12">
<!ATTLIST date month (%zero2twelve;) #REQUIRED>
<!ATTLIST date day (%zero2twelve;|13|14|15|16|17|18|19|20|21|22|23|24|25|26|27|28|29|30|31) #REQUIRED>
<!ATTLIST date year CDATA #REQUIRED>
<!ELEMENT cost (#PCDATA)>

<!--would hold data in a real xml file,
for now pretend it's in development-->
<!ELEMENT music (country,rap,pop)*>

```

```

<!ELEMENT film (comedy,drama)*>
<!ELEMENT country ANY>
<!ELEMENT rap ANY>
<!ELEMENT pop ANY>
<!ELEMENT comedy ANY>
<!ELEMENT drama ANY>

```

8. [8 marks] Consider the following XML fragment and give, for the elements in bold, their universal names and explain your choice:

```

1 <mns:name xmlns:mns="http://www.bear.org/"
2   xmlns:yms="http://www.bear.org/"
3   xmlns="http://www.wolf.org/">
4     <firstname>
5       <mns:nickname xmlns:mns="http://www.lamb.org/">
6     </mns:nickname >
7     <yms:nickname>
8     </yms:nickname >
9   </firstname>
10</mns:name>

```

- Line 1: `http://www.bear.org:/name` - from `mns:name` on line 1.
or `{http://www.bear.org/}name`
- Line 4: `http://www.wolf.org:/firstname` - from default namespace on line 3.
or `{http://www.wolf.org/}firstname`
- Line 5: `http://www.lamb.org:nickname` - from `mns:nickname` re-defined on line 5
or `{http://www.lamb.org}nickname`
- Line 7: `http://www.bear.org:/nickname` - from `yms:nickname` on line 2.
or `{ http://www.bear.org/}nickname`

COMP60362: SSD & XML Coursework

All written coursework is to be handed in to the postgraduate office by the following Monday, 11:30 am. In addition, please email XML document, DTDs, queries, xpaths, stylesheets, schemata, etc. to me at sattler@cs.man.ac.uk.

Files emailed are:

1. digitalmedia-stylesheet-1.xsl
2. digitalmedia-stylesheet-2.xsl
3. digitalmedia-stylesheet-3.xsl
4. digitalmedia-schema.xsd
5. MainDom.java
6. MainSax.java

9. [10 marks] What we have not discussed in detail so far are the different comparison operators provided by XPath to be used in predicates. Find out more about these operators and explain the node sets returned by each of the following XPath expressions, when executed on the example below:

- a. `/rootNode/childNode/deeperNode/T[text() = "t1"]`
- b. `/rootNode /childNode/deeperNode/T[text() eq "t1"]`
- c. `/rootNode/childNode/deeperNode[./T/text() eq "t1"]`
- d. `/rootNode/childNode /deeperNode[./T/text() = "t1"]`
- e. `/rootNode/childNode[deeperNode/T/text() = "t1"]`

```

<rootNode>
  <childNode>
    <deeperNode>
      <T>t1</T>
      <T>t2</T>
    </deeperNode>
    <deeperNode>
      <T>t1</T>
      <T>t3</T>
    </deeperNode>
    <deeperNode>
      <T>t2</T>
      <T>t5</T>
    </deeperNode>
  </childNode>
</rootNode>

```

The square brackets indicate a predicate, that is tested against the node list corresponding the search *before* the brackets. The `text()` simply matches all text nodes during the test. There are two operators in these queries. The “=” operator is comparing the string values of the members of the node set with the query term. It is a Boolean operation, and returns the node set which corresponds to true. The “eq” operator is a ‘value comparison’ operator. It is testing to see whether the nodes themselves are equal, in terms of their attributes and values, and the attributes and values of the sub-elements. In the comparison, the order of attributes is not important, since the operand is broken down by ‘atomization’ before the comparison is made. This is a ‘deep equals’.

a. `/rootNode/childNode/deeperNode/T[text() = "t1"]`

```

<T>t1</T>   - 4 -
<T>t1</T>   - 8 -

```

This XPath selects all the six `<T>` nodes, and among those then selects (by testing) the nodes with a text entry with a value of “t1”. Clearly, the only nodes selected will be as shown above. The operator = is making a string based equals comparison. The `<T>` nodes with a t2 or t3 or t5 text entry do not have the required text and so are not returned.

b. /rootNode/childNode/deeperNode/T[text() eq "t1"]

```
<T>t1</T> - 4 -
<T>t1</T> - 8 -
```

This is the same query with the operator replaced by eq. The nodes returned are the same. The difference lies in the operator. The *text* nodes have been selected as a node set (as above), and then the result of the test involves comparison to the *text* string "t1". The effect is that the same comparison is made as in part a.

c. /rootNode/childNode/deeperNode[./T/text() eq "t1"]

A sequence of more than one item is not allowed as the first operand of 'eq'

This query returns an error as shown above. The syntax of the "eq" operator has been violated. It is not possible to 'atomize' a sequence of more than one items, since the origin of any positive match could not be identified in the 'atomized' version.

d. /rootNode/childNode/deeperNode[./T/text() = "t1"]

```
<deeperNode> - 3 -
<T>t1</T> - 4 -
<T>t2</T> - 5 -
</deeperNode>

<deeperNode> - 7 -
<T>t1</T> - 8 -
<T>t3</T> - 9 -
</deeperNode>
```

This query tests the node set of <deeperNodes>. There is a "/" term which directs the search from the current node, since the "." is the actual <deeperNode>. Hence the nodes themselves (<deeperNode>), rather than the text nodes within them are selected. The path in the predicate brackets directs the comparison to the text <T> nodes, but the full node (with its contents) is reported for tests which return true. The comparison that is executed is between the text in the <T> nodes and the string "t1". Only the first two <deeperNode> nodes return true.

e. /rootNode/childNode[deeperNode/T/text() = "t1"]

```
<childNode> - 2 -
<deeperNode> - 3 -
  <T>t1</T> - 4 -
  <T>t2</T> - 5 -
</deeperNode> - 6 -
<deeperNode> - 7 -
  <T>t1</T> - 8 -
  <T>t3</T> - 9 -
</deeperNode> - 10 -
<deeperNode> - 11 -
  <T>t2</T> - 12 -
  <T>t5</T> - 13 -
</deeperNode> - 14 -
</childNode>
```

This query tests the single <childNode>. The test in the predicate is directed by the path to the <T> nodes. The query is asking whether the single child node has one or more examples of the form <deeperNode><T>t1</T></deeperNode>. Of course there are two examples, and so the query is true at the level of the single <childNode>, and this is returned by <Oxygen> with all its sub-elements.

10. [5 marks] For SAX and DOM, we talk a lot about "parse trees" and "DOM trees".

Provide a definition of a tree (in the CS sense) and reference its source.

Provide an example of a tree and an example of a graph that is not a tree.

For DOM, some people are concerned about the size of the DOM tree: how many nodes can a tree of depth

n have? You need to specify another parameter to answer this question.

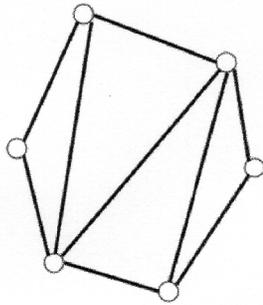
A graph is a model in which relationships between 'nodes' is represented by 'edges' that join the nodes.

"A tree is a connected undirected graph with no simple circuits."

"Since a tree cannot have a simple circuit, a tree cannot contain multiple edges or loops."

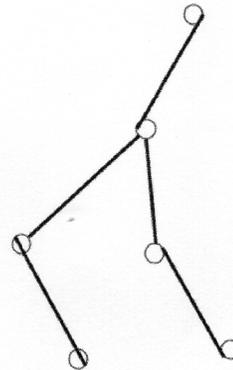
*"An undirected graph is a tree if and only if there is a **unique simple path** between any two of its vertices."*

(Ref: Discrete Mathematics and It's Applications (2002), Rosen,K., McGraw-Hill)



A graph that is Not a tree

A graph that Is a tree



In the figure on the left, there are cycles, and multiple paths to the same node. Hence the figure on the left is a graph, but not a tree.

(adapted from: Discrete Mathematics and It's Applications (2002), Rosen,K., McGraw-Hill)

To answer the second part of the question, it is necessary to define the number of child nodes that any node can have. If a binary tree is considered, with n levels, there are at least n nodes, and at most $2^{n+1}-1$ nodes (3 for 2 levels, 7 for 3 levels, 15 for 4 levels, etc), . Similar results can be obtained for other n-aries if the trees.

11. [14 marks] Make sure you have Java packages with DOM and SAX APIs (e.g., like the one available at <http://java.sun.com/webservices/jaxp/dist/1.1/docs/api/>). Use them to write two little Java programs that count the number of elements and the number of leaf elements in your XML document from Exercise 2 – one using a SAX parser, and a DOM parser. Explain your programs.

The code below uses the DOM api to determine that there are a total of 157 elements in my digitalmedia.xml file. Leaf elements are defined as elements that do not have other elements as child nodes (other types of child nodes, particularly text nodes, may be present). There are 101 leaf elements.

Summary of the code:

- A factory instance is used to build the XML parser
- The XML file is read and parsed. This creates a DOM tree with the root node at its head.
- The root node (Element) is found.
- A recursive method is used to process all the children, grand-children, etc of the root node.
- The result of the processing is printed.
- The recursive method (processAllChildren (Node)) examines all the child nodes of the input node (calling the processAllChildren() method on them), and determines how many Elements are present (adds Elements to an ArrayList and then counts them). It then determines the number of leaf elements by counting (i) the number of Elements with no children at all (ii) the number of Elements with no children that are instances of the Elements class.

```
package domANDsax;
import java.io.*;
```

```

import java.util.*;
import javax.xml.parsers.*;
import org.w3c.dom.*;
import org.xml.sax.*;
/**
 * Will run on Java 1.4 or Java 1.5
 * Question: Count the number of elements (including leaf elements) and
 *           count the number of leaf elements
 *           in digitalmedia.xml
 *           ... using DOM model.
 *
 * Result:
 * Total Number of Elements: 157
 * Number of Leaf Elements: 101
 *
 * Ref: Core Java Vol 2, Horstmann,C. and Cornell,G.
 * @K.J.Smith 2002-2007
 */
public class MainDom {
    private final boolean DEBUG = false;
    //hard-code the name of the file
    private final static String XMLfile = "/Users/kjsmith/Desktop/XML/domANDsax/digitalmedia.xml";

    private DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance ();
    private DocumentBuilder builderParser = null; {
        try{
            builderParser = factory.newDocumentBuilder ();
        } catch (ParserConfigurationException e){
            System.err.println (e.getMessage ());
        }
    }

    private ArrayList elementList = new ArrayList ();
    private ArrayList leafElementList = new ArrayList ();

    public MainDom (String xmlfile) {
        //read the xml file
        File file = new File (xmlfile);
        if (file.canRead () == false) { //error check
            System.out.println ("Unable to read .xml file.");
        }

        //build parse tree
        Document docParseTree = null;
        try {
            docParseTree = builderParser.parse (file);
        } catch (SAXException e) {
            System.err.println (e.getMessage ());
        } catch (IOException ioe) {
            System.err.println (ioe);
        }

        if (docParseTree == null) { //error check
            System.err.println ("Error. There is no parse tree.");
        }
    }

```

```

//-----
//start analysis
//get the root node
Element rootElement = null;
try {
    rootElement = docParseTree.getDocumentElement ();
} catch (NullPointerException npe) {
    System.err.println (npe.getMessage ());
}

if (rootElement == null) { //error check
    System.err.println ("Error. There is no root element.");
}

this.elementList.add (rootElement);
if (DEBUG) System.out.println ("Root Element is: " + rootElement.getNodeName ());

//recursively process all the nodes (of any type) of the root node
this.processAllChildren (rootElement);
//print out the final results
this.printResults ();
}

private void printResults () {
    System.out.println ("Total Number of Elements: " + this.elementList.size ());
    System.out.println ("Number of Leaf Elements: " + this.leafElementList.size ());

    //do some type checks just for completeness
    for (int i = 0; i < this.elementList.size (); i++) {
        if (!(this.elementList.get (i) instanceof Element)) {
            System.err.println ("Internal error in list of elements");
        }
    }

    for (int i = 0; i < this.leafElementList.size (); i++) {
        if (!(this.leafElementList.get (i) instanceof Element)) {
            System.err.println ("Internal error in list of leaf elements");
        }
    }

    if (DEBUG) for (int i = 0; i < this.leafElementList.size (); i++) {
        Element e = (Element)this.leafElementList.get (i);

        System.out.println ("xxx "+e.getNodeName ()
            + " xxx " +e.getParentNode ().getNodeName ()
            + " xxx " +e.getParentNode ().getNodeName ());
    }
}

//recursive method
private void processAllChildren (Node inNode) {
    if (inNode == null) {
        return;
    }
}

```

```

NodeList childNodes = this.getChildren (inNode);

if (childNodes == null) {
    return;
}

for (int i = 0; i < childNodes.getLength (); i++) {

    Node node = childNodes.item (i);
    if (node instanceof Element) {

        Element el = (Element)node;

        if (DEBUG) System.out.println (el.getNodeName () + " " + this.getChildrenCount (el));
        this.elementList.add (el);

        if (this.getChildrenCount (el) == 0) {
            this.leafElementList.add (el);
            continue;
        }

        NodeList nodelist = node.getChildNodes ();
        boolean elements = false;
        for (int j = 0; j < nodelist.getLength (); j++) {
            if (nodelist.item (j) instanceof Element) elements = true;
        }
        if (elements == false) {
            this.leafElementList.add (el);
        }
    }
}

if (this.getChildrenCount (node) == 0) {
    ; //terminate recursion
} else {
    this.processAllChildren (node); //recursive
}
}
}

//utility method
private NodeList getChildren (Node node) {
    if (node == null) {
        return null;
    }
    return node.getChildNodes ();
}

//utility method
private int getChildrenCount (Node node) {

    if (node == null) {
        return 0;
    }

    NodeList nodelist = this.getChildren (node);

```

```

    if (nodelist == null) {
        return 0;
    } else {
        return nodelist.getLength ();
    }
}

//-----

//run program from here
public static void main (String[] args) {
    if (args.length==0) {
        new MainDom (XMLfile);
    } else {
        new MainDom (args[0]);
    }
}
}

```

The following code uses the SAX2 API (in the Xerces package). The results are the same as above.

- A SAX2 parser is created and a handler (represented by the MainSax class which extends DefaultHandler) is associated.
- The XML file is read and parsed, and the results printed.
- The handler has a 'dummy' Element class to collect information about the Elements as they are passed over by the parser.
- The startElement() method creates a new Element object as each Element is opened. Information like the name is stored in the Element class. The level of the Element in the tree structure is recorded, and taken from the levelCurrent class attribute (this is incremented when an element is opened, and decremented when an Element is closed). The number of Elements is simply counted as the number of times the startElement() method is called. The Elements in the current branch of the tree are stored in the incrementingList class attribute (an ArrayList).
- The endElement() method represents the closing of Elements. It removes Elements from the list of elements in the current branch. The number of sub-elements for each Element in the current branch is measured here (an incrementing count is used). The number of leaf Elements is eventually determined by counting the number of Elements which have zero children.

```

package domANDSax;
import java.io.*;
import java.util.*;
import javax.xml.parsers.*;
import org.w3c.dom.*;
import org.xml.sax.*;
import org.xml.sax.helpers.*;
/**
 * Will run on Java 1.4 or Java 1.5
 * Question: Count the number of elements (including leaf elements) and
 *           count the number of leaf elements
 *           in digitalmedia.xml
 *           ... using SAX model - from SAX2.
 *
 * Result:
 * Total Number of Elements: 157
 * Number of Leaf Elements: 101

```

```

*
* Ref: Core Java Vol 2, Horstmann,C. and Cornell,G.
* Ref:http://www.javaworld.com/javaworld/jw-08-2000/jw-0804-sax.html
* @K.J.Smith 2002-2007
*/
public class MainSax extends DefaultHandler {
    private final boolean DEBUG = false;
    private int levelCurrent = 0; //indicates the level within the tree
    //hard-code the name of the file
    private final static String XMLfile = "/Users/kjsmith/Desktop/XML/domANDsax/digitalmedia.xml";

    private ArrayList incrementingList = new ArrayList (); //contains the Elements in the current branch
    private ArrayList elementList = new ArrayList (); // contains all Elements encountered

    private class Element { // a class for storing thins about Elements
        private int noOfChildren = 0;
        private int level = -1;
        private String contents = null;
        private String name = null;
    }
    private boolean trigger = false; //used to collect the contents of the Element from contentBuffer
    private CharArrayWriter contentsBuffer = new CharArrayWriter (); // Buffer for collecting data from
    SAX event

    public void startElement ( String namespaceURI, String localName, String qName, Attributes attr )
    throws SAXException {
        contentsBuffer.reset ();

        this.levelCurrent++; //opening a new Element so it must be a new level
        Element e = new Element (); //a new Element has been encountered so build an object for it
        e.name = localName; //set the name of the new element
        this.trigger = true; //trigger collection of the contents from the content Buffer
        e.level=this.levelCurrent; //record the level of the element
        this.elementList.add (e); //add the element to the total list
        this.incrementingList.add (e); //add the element to the list for the current branch

        this.startElements++; //increment the count of elements opened
        if (attr != null) {
            this.startAttributes += attr.getLength ();
        }

        this.printStart (namespaceURI, localName, qName, attr); //debug
    }
    //debugging
    private void printStart (String namespaceURI, String localName, String qName, Attributes attr) {
        if (!DEBUG) return;

        if (namespaceURI.trim ().length () > 0) {
            System.out.println ("namespaceURI: " + namespaceURI);
        }
        if (localName.trim ().length () > 0) {
            System.out.println ("localName: " + localName);
        }
        if (qName.trim ().length () > 0) {
            System.out.println ("qName: " + qName);
        }
    }
}

```

```

    if (attr != null) {
        for (int i = 0; i < attr.getLength (); i++) {
            System.out.println ("+Attribute(" + i + "): " + attr.getLocalName (i)
                + " " + attr.getType (i) + " " + attr.getValue (i));
        }
    }
    System.out.println ("+++++" + this.levelCurrent);
}
public void endElement ( String namespaceURI, String localName, String qName )
throws SAXException {
    this.levelCurrent--; //closing an element so decrement the level
    this.endElements++; //count the number of elements closed
    this.printEnd (namespaceURI, localName, qName); //debug
    this.removeCurrentLevel (); //remove the old element from the current branch
    this.incrementCurrent (); //record the number of subelements (children. grand children etc)
    this.updateCurrentElement (); //complex elements will have a tag structure in contents - so fix it
}
private void removeCurrentLevel () { //remove the old element from the current branch
    for (int i = 0; i < this.incrementingList.size (); i++) {
        Element e = (Element) this.incrementingList.get (i);
        if (e.level > this.levelCurrent) {
            this.incrementingList.remove (e);
        }
    }
}
private void incrementCurrent () { //record the number of subelements (children. grand children etc)
    for (int i = 0; i < this.incrementingList.size (); i++) {
        Element e = (Element) this.incrementingList.get (i);
        e.noOfChildren++;
    }
}
private void updateCurrentElement () { //complex elements will have a tag structure in contents - so fix it
    for (int i = 0; i < this.incrementingList.size (); i++) {
        Element e = (Element) this.incrementingList.get (i);

        if (e.level == this.levelCurrent) {
            if (this.contentsBuffer.toString ().trim ().length () > 0) {
                e.contents=this.contentsBuffer.toString ();
            }
        }
    }
}
//debugging
private void printEnd (String namespaceURI, String localName, String qName) {
    if (!DEBUG) return;

    if (namespaceURI.trim ().length () > 0) {
        System.out.println ("-namespaceURI: " + namespaceURI);
    }
    if (localName.trim ().length () > 0) {
        System.out.println ("-localName: " + localName);
    }
    if (qName.trim ().length () > 0) {
        System.out.println ("-qName: " + qName);
    }
}

```

```

    if ((contentsBuffer != null) && (contentsBuffer.toString().trim().length() > 0)) {
        System.out.println ("-Contents:" + contentsBuffer.toString());
    }
    System.out.println ("-----" + this.levelCurrent);
}
public void characters ( char[] ch, int start, int length )
throws SAXException {
    this.contentsBuffer.write ( ch, start, length ); // accumulate the contents into a buffer
    if (this.trigger == true) update ();
    this.trigger = false;
}
private void update () { //copy contentsBuffer into current (last in list) Element
    int end = this.elementList.size ();
    Element e = (Element) this.elementList.get (end - 1);
    if (this.contentsBuffer.toString().trim().length() > 0) {
        e.contents=this.contentsBuffer.toString ();
    }
}
private int startElements = 0; //count of the number of elements opened
private int endElements = 0; //count of the elements closed
private int startAttributes = 0; //count of the total number of attributes

public void printSizes () { //print the results
    System.out.println ("EndElements: " + this.endElements);
    System.out.println ("StartElements: " + this.startElements);
    System.out.println ("StartAttributes: " + this.startAttributes);

    if (DEBUG) System.out.println (this.incrementingList.size () + " " + this.elementList.size ());

    if (DEBUG) for (int i = 0; i < this.elementList.size (); i++) {
        Element e = (Element) this.elementList.get (i);
        System.out.println (e.noOfChildren+ " "+###"+e.contents+"###"+e.name);
    }

    int count = 0;
    for (int i = 0; i < this.elementList.size (); i++) {
        Element e = (Element) this.elementList.get (i);

        if (e.noOfChildren == 0) {
            count++;
            if (DEBUG) System.out.println ("***"+e.contents+"***" + e.name);
        }
    }

    System.out.println ("Total Number of Elements: " + this.startElements);
    System.out.println ("Number of Leaf Elements: " + count);
}

//-----
public MainSax (String xmlfile) {
    //read the xml file
    File file = new File (xmlfile);
    if (file.canRead () == false) { //error check
        System.out.println ("Unable to read .xml file.");
    }
}

```

```

try {
    // specify SAX2 parser
    XMLReader xr = XMLReaderFactory.createXMLReader ("org.apache.xerces.parsers.SAXParser");
    // The rest of the class is a ContentHandler
    xr.setContentHandler ( this );
    // do Parsing
    xr.parse ( new InputSource ( new FileReader ( XMLfile ) ) );

    this.printSizes ();

} catch ( SAXException se ) {
    se.printStackTrace ();
} catch ( FileNotFoundException fe ) {
    fe.printStackTrace ();
} catch ( IOException ioe ) {
    ioe.printStackTrace ();
}
}
//-----
//run program from here
public static void main (String[] args) {
    if (args.length==0) {
        new MainSax (XMLfile);
    } else {
        new MainSax (args[0]);
    }
}
}

```

12. [5 marks] Describe in 5 sentences what a “Turing complete programming language” is.

- In general terms, a programming language is Turing complete if it is capable of simulating the behavior of other Turing Programming languages, irrespective of how difficult the simulation is to create or how many resources are required to execute it.
- There are various views of the minimum functionality that is required to define a Turing programming language, for example if it “*has integer variables, arithmetic and sequentially executes statements, which include assignment, selection (if) and loop (while) statements*”. (Ref: <http://www.mcs.drexel.edu/~jjohnson/2006-07/winter/cs360/lectures/lec1.html>)
- If a Turing machine is a theoretical computer that can execute *any* well- defined algorithm, and a Turing complete program is the sequence of steps performed to execute that arbitrary algorithm, then a Turing complete programming language supports the execution of that program.
- It is difficult to find a programming language (that is of general utility) and is not Turing complete (provided you ignore implementation dependent resource limitations).
- One result in the theory of computation, is the halting problem, which can be defined to say that it is impossible to decide whether a program written in a Turing complete program language will ever terminate – so that implementations of languages that allow termination after a maximum computation time are strictly not Turing complete.

13. [12 marks] Use <oXygen/> to write and test the following stylesheets for your XML document from Exercise 2.

- a. Write a stylesheet that transforms your XML document (including some attribute values) into another well-formed XML document which uses different element names and which is of a different, more **shallow** structure.

<?xml-stylesheet type="text/xsl" href="digitalmedia-stylesheet-3.xsl"?> is added to associate a stylesheet. Saxon was used as the XSLT processor.

-This stylesheet takes the input digitalmedia.xml file and performs the following operations to create a new XML file.

The output mode is XML

- A lot of information is first filtered out and thrown away (e.g. music, films, and documentaries), leaving only plays and books.
- Some information from plays and books is discarded.
- The aim of the remainder of the stylesheet is to reformat the play and book structures to be similar.
- <book> is renamed <audiobook>
- The genre tag is promoted from attribute to element.
- The <author> is lifted upwards out of the <publication-details> tag
- The name of <format> is changed to <file> and the <type> tag is promoted from attribute to element
- The format of the <play> tag (renamed to <radioplay>) is modified to match the format of the <audiobook> in a similar way.

```

<?xml version="1.0" encoding="UTF-8"?>
<!--
  Builds a XML document digitalmedia-stylesheet-2.xsl
-->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml"/>

  <!--filter out-->
  <xsl:template match="digitalmedia/music"/>
  <xsl:template match="digitalmedia/film"/>
  <xsl:template match="digitalmedia/spokenword/documentary"/>

  <xsl:template match="digitalmedia/spokenword/book/actor"/>
  <xsl:template match="digitalmedia/spokenword/book/info"/>
  <xsl:template match="digitalmedia/spokenword/book/publication_detail"/>
  <xsl:template match="digitalmedia/spokenword/book/link"/>
  <xsl:template match="digitalmedia/spokenword/book/format"/>

  <xsl:template match="digitalmedia/spokenword/play/format"/>
  <xsl:template match="digitalmedia/spokenword/play/link"/>
  <xsl:template match="digitalmedia/spokenword/play/actor"/>
  <xsl:template match="digitalmedia/spokenword/play/info"/>

  <xsl:template match="digitalmedia/spokenword/book">
    <audiobook>
      <!-- promote genre from attribute to tag-->
      <xsl:element name="genre">
        <xsl:value-of select="attribute::genre"/>
      </xsl:element>
      <!--lift the level of author-->
      <author>
        <xsl:copy-of select="publication_detail/author/name"/>
      </author>

      <file>
        <!-- promote type from attribute to tag-->
        <xsl:element name="type">
          <xsl:value-of select="format/attribute::type"/>
        </xsl:element>
      </file>
    </audiobook>
  </xsl:template>
</xsl:stylesheet>

```

Discard

Promote

wft

Promote

`</xsl:element>`
`<xsl:copy-of select="format/*"/>`
`</file>`

`<xsl:apply-templates/>`
`</audiobook>`
`</xsl:template>`

`<xsl:template match="digitalmedia/spokenword/play">`
`<radioplay>`
`<!-- promote genre from attribute to tag-->`
`<xsl:element name="genre">`
`<xsl:value-of select="attribute::genre"/>`
`</xsl:element>`

`<file>`
`<!-- promote type from attribute to tag-->`
`<xsl:element name="type">`
`<xsl:value-of select="format/attribute::type"/>`
`</xsl:element>`
`<xsl:copy-of select="format/*"/>`
`</file>`

`<xsl:apply-templates/>`
`</radioplay>`
`</xsl:template>`

`<xsl:template match="@*|node()">`
`<xsl:copy>`
`<xsl:apply-templates select="@*|node()"/>`
`</xsl:copy>`
`</xsl:template>`

`</xsl:stylesheet>`

The output xml file is shown below.

```

<?xml version="1.0" encoding="utf-8"?><!--
K. John Smith
Exercise 2
--><!-- The following DTD of the types internal and external
added for Exercise 4-->
<digitalmedia>
  <spokenword>

    <audiobook>
      <genre>drama</genre>
      <author>
        <name>
          <family>London</family>
          <given>Jack</given>
        </name>
      </author>
      <file>
        <type>audible</type>
        <location>http://macHome9/media/audible/seawolf</location>
        <source>

```

```

    <owner>©Audible.com</owner>
    <location type="incomplete">Audible.com</location>
    <cost>$7.49</cost>
  </source>
</file>
<title>The Sea Wolf</title>
</audiobook>

<audiobook>
  <genre>drama</genre>
  <author>
    <name>
      <family>Melville</family>
      <given>Herman</given>
    </name>
  </author>
  <file>
    <type>audible</type>
    <location>http://macHome9/media/audible/bartleby/</location>
    <source>
      <owner>©Audible.co.uk</owner>
      <location type="incomplete">Audible.co.uk</location>
      <cost>£9.99</cost>
    </source>
  </file>
  <title>Bartleby the Scrivener, and other stories <subtitle>Bartleby the
  Scrivener</subtitle>
    <subtitle>The Lightning-Rod Man</subtitle>
    <subtitle>The Bell-Tower</subtitle>
  </title>
</audiobook>

<radioplay>
  <genre>drama</genre>
  <file>
    <type>mp3</type>
    <location>http://macHome9/media/radio/inspectorcalls.mp3</location>
    <source>
      <owner>©bbc</owner>
      <location type="incomplete">bbc7</location>
      <cost>0</cost>
    </source>
  </file>
  <title>An Inspector Calls</title>
  <author>
    <name>
      <family>Priestley</family>
      <given>J.B.</given>
    </name>
    <info>http://en.wikipedia.org/wiki/J._B._Priestley</info>
  </author>
</radioplay>

<radioplay>
  <genre>drama</genre>
  <file>

```

```

<type>mp3</type>
<location>http://macHome9/media/radio/nightmother.mp3</location>
<source>
  <owner>©bbc</owner>
  <location type="incomplete">bbc7</location>
  <cost>0</cost>
</source>
</file>
<title>'night, Mother</title>
<author>
  <name>
    <family>Norman</family>
    <given>Marsha</given>
  </name>
</author>
</radioplay>

<radioplay>
  <genre>drama</genre>
  <file>
    <type>ra</type>
    <location>http://macHome9/media/radio/breaker.mp3</location>
    <source>
      <owner>©bbc</owner>
      <location type="incomplete">bbc radio4</location>
      <cost>0</cost>
    </source>
  </file>
  <title>Breaker Morant</title>
  <author>
    <name>
      <given>Kenneth</given>
      <family>Ross</family>
    </name>
  </author>
</radioplay>

</spokenword>

</digitalmedia>

```

b. Write a stylesheet that transforms your XML document into a well-structured, readable **HTML** document.

The stylesheet is designed to form the HTML page illustrated. It includes...

- Standard html, title and body tags. The output mode is html.
- A simple list of titles of plays, books, and documentaries is presented, sorted by the alphabetical order of the title, with the subtitle handled separately and placed in brackets. Prioritized templates, which select based on the presence of a subtitle (as highest priority) as used. A comma is added between subtitles, excepting the last subtitle in the list.
- A table is formed with appropriate headings.
- Key points are ... the title, which has the subtitle discarded.
- Format is taken from an attribute.
- Author is taken from the different formats in books as opposed to plays/documentaries
- Genre is extracted from book|play, but is manually inserted for documentary.

- The author which is formed from the given and lastnames, and may contain a list (a for-each used).
- The location which is provided as a hyperlink using the tag format

```

<?xml version="1.0" encoding="UTF-8"?>
<!--
  Builds a web page. digitalmedia-style-sheet-3.xsl
-->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:output method="html"/>

  <xsl:template match="/">

    <html>
      <title>Digital Media Library</title>
      <head/>
      <body>
        <ol>
          <xsl:apply-templates select="/digitalmedia/spokenword/*/title" mode="o">
            <xsl:sort select="." data-type="text" order="ascending"/>
          </xsl:apply-templates>
        </ol>

        <br/>

        <table cellspacing="0" cellpadding="0" border="1">
          <tbody>
            <tr>
              <th>Classification</th>
              <th>Title</th>
              <th>Author</th>
              <th>Format</th>
              <th>Location</th>
              <th>Source</th>
            </tr>

            <xsl:for-each select="/digitalmedia/spokenword/*">
              <tr>
                <td>
                  <xsl:apply-templates select="." mode="c"/>
                </td>
                <td>
                  <xsl:apply-templates select="title"/>
                </td>
                <td>
                  <xsl:apply-templates select="publication_detail/author"/>
                  <xsl:apply-templates select="author"/>
                </td>
                <td>
                  <xsl:value-of select="format/@type"/>
                </td>
                <td>
                  <A>
                    <xsl:attribute name="HREF">
                      <xsl:value-of select="format/location"/>

```

```

        </xsl:attribute>
        <xsl:value-of select="format/location"/>
    </A>
</td>
<td>
    <xsl:value-of select="format/source/owner"/>
</td>
</tr>
</xsl:for-each>
</tbody>
</table>
</body>
</html>
</xsl:template>

<xsl:template match="title[subtitle]" mode="o" priority="0.5">
<li>
    <xsl:value-of select="../title/node()"/> ( <xsl:for-each select="./subtitle">
        <xsl:value-of select="."/>
        <xsl:if test="position()=last()">, </xsl:if>
    </xsl:for-each> ) </li>
</xsl:template>

<xsl:template match="title" mode="o" priority="0">
<li>
    <xsl:value-of select="."/>
</li>
</xsl:template>

<xsl:template match="title[subtitle]" priority="0.5">
    <xsl:value-of select="../title/node()"/>
</xsl:template>

<xsl:template match="title" priority="0">
    <xsl:value-of select="."/>
</xsl:template>

<xsl:template match="book|play" mode="c">
    <xsl:value-of select="./@genre"/>
</xsl:template>

<xsl:template match="documentary" mode="c">
    <xsl:text>documentary</xsl:text>
</xsl:template>

<xsl:template match="author">
    <xsl:for-each select="name">
        <xsl:value-of select="given"/>
        <xsl:text> </xsl:text>
        <xsl:value-of select="family"/>
        <br/>
    </xsl:for-each>
</xsl:template>
</xsl:stylesheet>

```

Getting Started Latest Headlines

1. 'night, Mother
2. An Inspector Calls
3. Bartleby the Scrivener, and other stories (Bartleby the Scrivener, The Lightning-Rod Man, The Bell-Tower)
4. Before It Had a Name (TAL Episode 197)
5. Breaker Morant
6. The Sea Wolf
7. Trail of Tears (TAL Episode 107)

Classification	Title	Author	Format	Location	Source
drama	The Sea Wolf	Jack London	audible	http://macHome9/media/audible/seawolf/	©Audible.com
drama	Bartleby the Scrivener, and other stories	Herman Melville	audible	http://macHome9/media/audible/bartleby/	©Audible.co.uk
drama	An Inspector Calls	J.B. Priestley	mp3	http://macHome9/media/radio/inspectorcalls.mp3	©bbc
drama	'night, Mother	Marsha Norman	mp3	http://macHome9/media/radio/nightmother.mp3	©bbc
drama	Breaker Morant	Kenneth Ross	ra	http://macHome9/media/radio/breaker.mp3	©bbc
documentary	Trail of Tears	Ira Glass Sarah Vowell	mp3	http://macHome9/media/TAL/107.mp3	©PBS
documentary	Before It Had a Name	Ira Glass	mp3	http://macHome9/media/TAL/197.mp3	©PBS

c. Write a stylesheet that transforms your XML document into a **flat file** where space and linebreaks are the only means of structuring.
Explain your stylesheets.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  Builds a flat file, digitalmedia-stylesheet-1.xsl
-->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:output method="text"/>

  <xsl:template match="/">

    <xsl:text>Classification Title Author_FirstName Author_LastName Format Location
    Source</xsl:text>
    <xsl:text>&#10;</xsl:text>
    <xsl:for-each select="/digitalmedia/spokenword/*">
      <xsl:apply-templates select="." mode="c"/>
      <xsl:apply-templates select="title"/>
      <xsl:apply-templates select="publication_detail/author"/>
      <xsl:apply-templates select="author"/>
      <xsl:value-of select="format/@type"/>
      <xsl:text> </xsl:text>
      <xsl:value-of select="format/location"/>
      <xsl:text> </xsl:text>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

```

<xsl:value-of select="format/source/owner"/>
<xsl:text> </xsl:text>
<xsl:text>&#10;</xsl:text>

</xsl:for-each>
</xsl:template>

<xsl:template match="title[subtitle]" priority="0.5">
<xsl:text>"</xsl:text>
<xsl:value-of select="."/title/node()"/>
<xsl:text>"</xsl:text>
<xsl:text> </xsl:text>
</xsl:template>

<xsl:template match="title" priority="0">
<xsl:text>"</xsl:text>
<xsl:value-of select="."/>
<xsl:text>"</xsl:text>
<xsl:text> </xsl:text>
</xsl:template>

<xsl:template match="book|play" mode="c">
<xsl:value-of select="."/genre"/>
<xsl:text> </xsl:text>
</xsl:template>

<xsl:template match="documentary" mode="c">
<xsl:text>documentary</xsl:text>
<xsl:text> </xsl:text>
</xsl:template>

<xsl:template match="author">
<xsl:for-each select="name">
<xsl:value-of select="given"/>
<xsl:text> </xsl:text>
<xsl:value-of select="family"/>
<xsl:text> </xsl:text>
</xsl:for-each>
</xsl:template>

</xsl:stylesheet>

```

- A flat file is output, the intension being to mimic the information in the web page. Much of the code was copied form the web page transformation code.
- The output mode is text.
- Linebreaks are included as
 and spaces using <xsl:text> </xsl:text>
- Since the title can include spaces within itself, " " quotes are added around it, as a delimeter.
- First and last names are treated separately.

The output is shown below...

Classification	Title	Author_FirstName	Author_LastName	Format	Location	Source
drama	"The Sea Wolf"	Jack	London	audible	http://macHome9/media/audible/seawolf/	©Audible.com
drama	"Bartleby the Scrivener, and other stories"	Herman	Melville	audible	http://macHome9/media/audible/bartleby/	©Audible.co.uk
drama	"An Inspector Calls"	J.B.	Priestley	mp3	http://macHome9/media/radio/inspectorcalls.mp3	©bbc
drama	"night, Mother"	Marsha	Norman	mp3	http://macHome9/media/radio/nightmother.mp3	©bbc

drama "Breaker Morant" Kenneth Ross ra <http://macHome9/media/radio/breaker.mp3> ©bbc
 documentary "Trail of Tears " Ira Glass Sarah Vowell mp3 <http://macHome9/media/TAL/107.mp3> ©PBS
 documentary "Before It Had a Name " Ira Glass mp3 <http://macHome9/media/TAL/197.mp3> ©PBS

14. [12 marks] Use <oxygen/> to write an XML Schema for your XML document from Exercise 2, and make sure that your document is indeed valid w.r.t. this schema. Your Schema should

- contain at least four named types, among them simple ones and complex ones,
- make use of both xs:restriction and xs:extension,
- make use of different datatypes, and
- make use of at least two different element enforcement constructors and the attributes minOccurs and maxOccurs.

Explain your schema.

The schema follows closely the layout of the dtd. Key points are...

- in <sequence> and <all> (see nameType), restrictions are placed on the minOccurs and maxOccurs counts of elements included in many elements. These correspond to the values in the dtds (defined by *, +, etc).
- mediaType is a complexType base extended by book, play, and documentary
- textEntry is a simpleType used to define string datatypes of upto 80 characters,
- Title is a mixed complex type since it includes text and subtitle elements
- PersonTypes are extended from shortPersonTypes. The former is used for author, the later for actor
- Info is a mixed complex type because it has text and attributes
- Link has a required IDREF datatype
- Name is an example with attributes, and elements, and uses the nameType complex type.
- Format uses an enumerated list restriction, as does genretype and locationList.
- Types are placed at the end of the document in order to help with legibility.

The xml file needs to incorporate

<digitalmedia xmlns:xsi=<http://www.w3.org/2001/XMLSchema-instance>
 xsi:noNamespaceSchemaLocation="digitalmedia-schema.xsd">. It is then proven to be valid with respect to the schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <!-- Activate with
  <digitalmedia xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="digitalmedia-schema.xsd" -->

  <xs:element name="digitalmedia">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="spokenword" minOccurs="1" maxOccurs="unbounded"/>
        <xs:element ref="music" minOccurs="1" maxOccurs="unbounded"/>
        <xs:element ref="film" minOccurs="1" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="spokenword">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="book" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="play" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```

    <xs:element ref="documentary" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
</xs:element>

<xs:element name="book">
  <xs:annotation>
    <xs:documentation>A book.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="mediaType">
        <xs:sequence>
          <xs:element ref="title" minOccurs="1" maxOccurs="1"/>
          <xs:element ref="publication_detail" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="format" minOccurs="1" maxOccurs="unbounded"/>
          <xs:element ref="actor" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="info" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="link" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>

<xs:element name="documentary">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="mediaType">
        <xs:sequence>
          <xs:element ref="title" minOccurs="1" maxOccurs="1"/>
          <xs:element ref="author" minOccurs="1" maxOccurs="unbounded"/>
          <xs:element ref="date" minOccurs="0" maxOccurs="1"/>
          <xs:element ref="format" minOccurs="1" maxOccurs="unbounded"/>
          <xs:element ref="actor" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="info" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="link" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>

<xs:element name="play">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="mediaType">
        <xs:sequence>
          <xs:element ref="title" minOccurs="1" maxOccurs="1"/>
          <xs:element ref="author" minOccurs="1" maxOccurs="unbounded"/>
          <xs:element ref="format" minOccurs="1" maxOccurs="unbounded"/>
          <xs:element ref="actor" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="info" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="link" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>

```

```

    </xs:complexContent>
  </xs:complexType>
</xs:element>

<xs:element name="publication_detail">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="author" minOccurs="1" maxOccurs="unbounded"/>
      <xs:element ref="edition" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="publisher" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="edition" type="textEntry"/>
<xs:element name="publisher" type="textEntry"/>

<xs:element name="title">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element ref="subtitle" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="subtitle" type="textEntry"/>

<xs:element name="author" type="personType"/>

<xs:element name="format">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="location" minOccurs="1" maxOccurs="unbounded"/>
      <xs:element ref="source" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="type" type="formatList"/>
  </xs:complexType>
</xs:element>

<xs:element name="actor" type="shortPersonType"/>

<xs:element name="info">
  <xs:complexType mixed="true">
    <xs:attribute name="title" type="textEntry"/>
  </xs:complexType>
</xs:element>

<xs:element name="link">
  <xs:complexType mixed="true">
    <xs:attribute name="related" type="xs:IDREF" use="required"/>
  </xs:complexType>
</xs:element>

<xs:complexType name="name">
  <xs:sequence>
    <xs:element ref="title" minOccurs="1" maxOccurs="1"/>
    <xs:element ref="author" minOccurs="1" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

```

```

</xs:sequence>
  <xs:attribute name="genre" type="genreList" default="misc"/>
  <xs:attribute name="id" type="xs:ID" use="required"/>
</xs:complexType>

<xs:element name="family" type="xs:string"/>
<xs:element name="given" type="xs:string"/>

<xs:element name="picture" type="textEntry"/>

<xs:element name="source">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="owner" minOccurs="1" maxOccurs="1"/>
      <xs:element ref="location" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="date" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="cost" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="location">
  <xs:complexType mixed="true">
    <xs:attribute name="type" type="locationList" default="incomplete"/>
  </xs:complexType>
</xs:element>

<xs:element name="owner" type="textEntry"/>
<xs:element name="date"/>
<xs:element name="cost" type="textEntry"/>

<!-- not being used at the moment -->
<xs:element name="music"/>
<xs:element name="film"/>
<xs:element name="country"/>
<xs:element name="rap"/>
<xs:element name="pop"/>
<xs:element name="comedy"/>
<xs:element name="drama"/>

<!-- xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx -->

<xs:complexType name="mediaType">
  <xs:attribute name="genre" type="genreList" default="misc"/>
  <xs:attribute name="id" type="xs:ID" use="required"/>
</xs:complexType>

<xs:complexType name="personType">
  <xs:complexContent>
    <xs:extension base="shortPersonType">
      <xs:sequence>
        <xs:element ref="picture" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

```

```

<xs:complexType name="shortPersonType">
  <xs:sequence>
    <xs:element name="name" type="nameType" minOccurs="1" maxOccurs="1"/>
    <xs:element ref="info" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="nameType">
  <xs:all>
    <xs:element ref="family" minOccurs="1" maxOccurs="1"/>
    <xs:element ref="given" minOccurs="1" maxOccurs="1"/>
  </xs:all>
</xs:complexType>

<xs:simpleType name="formatList">
  <xs:restriction base="xs:string">
    <xs:enumeration value="mp3"/>
    <xs:enumeration value="flv"/>
    <xs:enumeration value="audible"/>
    <xs:enumeration value="FlashPlayer9"/>
    <xs:enumeration value="ra"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="genreList">
  <xs:restriction base="xs:string">
    <xs:enumeration value="drama"/>
    <xs:enumeration value="misc"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="locationList">
  <xs:restriction base="xs:string">
    <xs:enumeration value="complete"/>
    <xs:enumeration value="incomplete"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="textEntry">
  <xs:restriction base="xs:string">
    <xs:minLength value="0"/>
    <xs:maxLength value="80"/>
  </xs:restriction>
</xs:simpleType>

</xs:schema>

```

15. [12 marks] Use <oxygen/> to write XQueries for your XML document from Exercise 2.
- Write an XQuery that generates the same output as the stylesheet you wrote for Exercise 13.a.
 - Write an XQuery that generates the same output as the stylesheet you wrote for Exercise 13.b.
 - Write an XQuery that generates the same output as the stylesheet you wrote for Exercise 13.c.
- Explain your queries.

16. [9 marks] In 3 sentences, explain in what kind of situation you would rather use XSLT than XQuery and why. In 3 sentences, explain in what kind of situation you would rather use XQuery than XSLT and why.

XQuery rather than XSLT:

- XQuery is best suited to processing data, for example, it is used to query XML in a database or quasi-database context.
- XQuery is supported by databases like Oracle, and has an SQL-like FLWOR syntax, so that queries can 'order by' or aggregate the data, and so is particularly suited to these kinds of analyses.
- Where strongly typed (compile time) data checks are required, for example in a database query context, then XQuery is preferred since XSLT is only weakly typed.

XSLT rather than XQuery:

- XSLT is best suited to processing documents, for example the transformation of XML to XHTML, for presentation in a web browser.
- For processing very large files, the indexing capability (using key) in XSLT can dramatically improve performance, but this capability is absent from XQuery.
- XSLT is an established technology very widely supported, for example in web browsers, which provide XSLT processors, and so should be used for better compatibility in contexts where the technology available to the end user is not known.