From Introductory Lecture

1. Core Concepts

ORGANISATION- How data is represented/associated

METADATA Data about what the data is

ACCESS How get at the data efficiently

2. Common Challenges for enterprises and individuals

Volume
Velocity
Variety
Validity (including Ethics)

Next thing to consider - How to use and extract value from the data- leading to the semantic web lecture on November 26th

Quick Quiz

1. Three types of Markup Language are
XML, EDIFACT and ______

2. The Acrynom DTD stands for
__________ ___________ ___________

3. DTDs and XML Schemas are used to _________ XML files.

4. DTD and XML Schemas can be used to foster community ________ on data models.

5. An XML document can be represented as a _____ structure

6. Xpath is used to ________ an XML document

7. The acrynom FLWOR for Xqueries stands for ______,______, ______,______, ______

8. BaseX is an XML __________

9. DTDs do not support _________

10. XSL stands for eXtensible Stylesheet Language. What does XSLT stand for?________________________

Quick Quiz- with Suggested Answers

1. Three types of Markup Language are
XML, EDIFACT and **HTML**

2. The Acrynom DTD stands for **Document Type Definition**

3. DTDs and XML Schemas are used to **Validate** XML files.

4. DTD and XML Schemas can be used to foster community **Consensus** on data models.

5. An XML document can be represented as a **Tree** structure

6. Xpath is used to **Navigate** an XML document

7. The acrynom FLWR for Xqueries stands for **For**, **Let**, **Where**, **Order by**, **Return**

8. BaseX is an XML **Processor**

9. DTDs do not support **Namespaces**

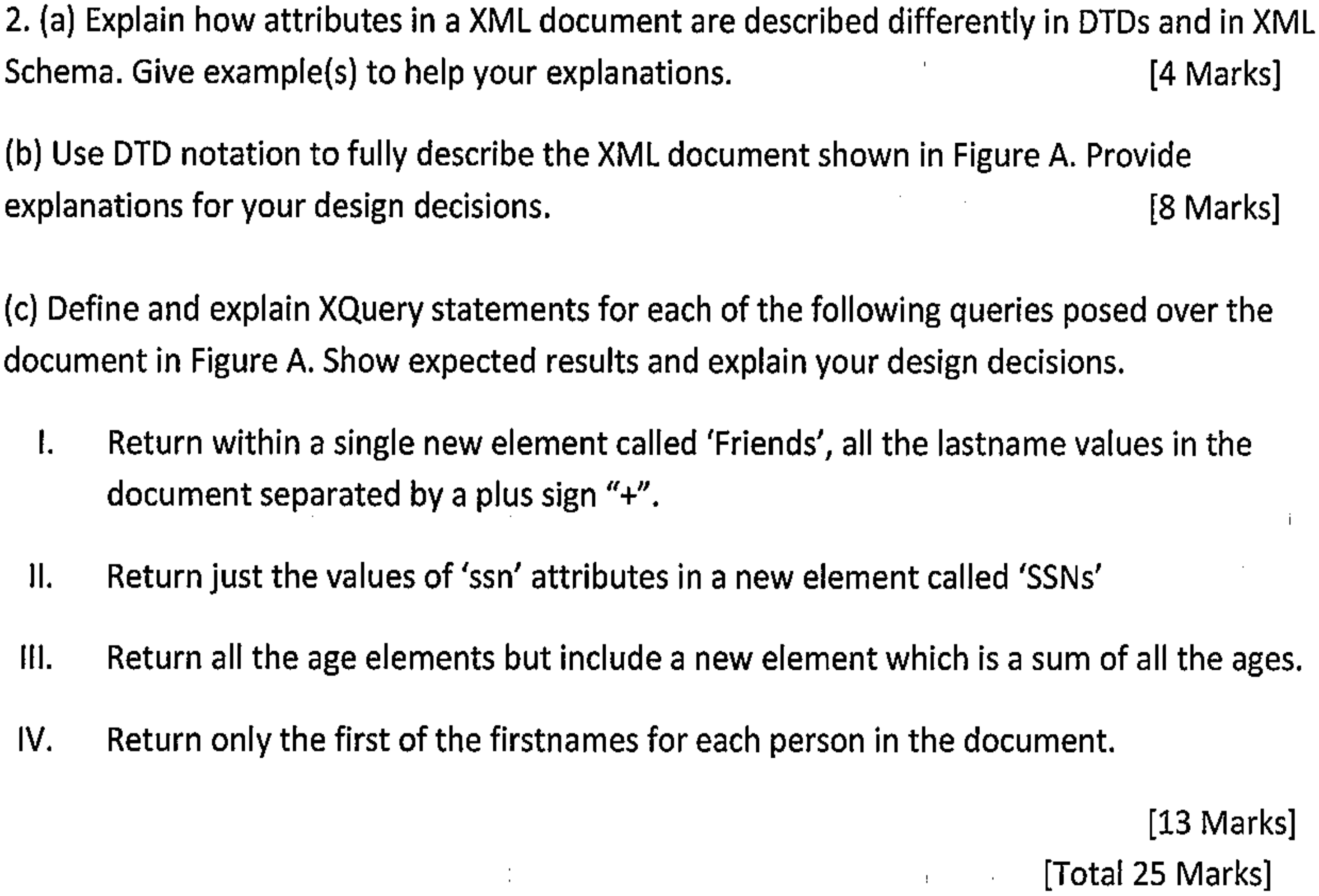
10. XSL stands for eXtensible Stylesheet Language. What does XSLT stand for?
eXtensible Stylesheet Language Transformation

Change to Assignment Submission Date

- **Change to assignment submission Date**

- the new submission date for your assignment is Monday November 26th.

Past XML Exam Question



2. (a) Explain how attributes in a XML document are described differently in DTDs and in XML Schema. Give example(s) to help your explanations. [4 Marks]

(b) Use DTD notation to fully describe the XML document shown in Figure A. Provide explanations for your design decisions. [8 Marks]

(c) Define and explain XQuery statements for each of the following queries posed over the document in Figure A. Show expected results and explain your design decisions.

I. Return within a single new element called 'Friends', all the lastname values in the document separated by a plus sign "+".

II. Return just the values of 'ssn' attributes in a new element called 'SSNs'

III. Return all the age elements but include a new element which is a sum of all the ages.

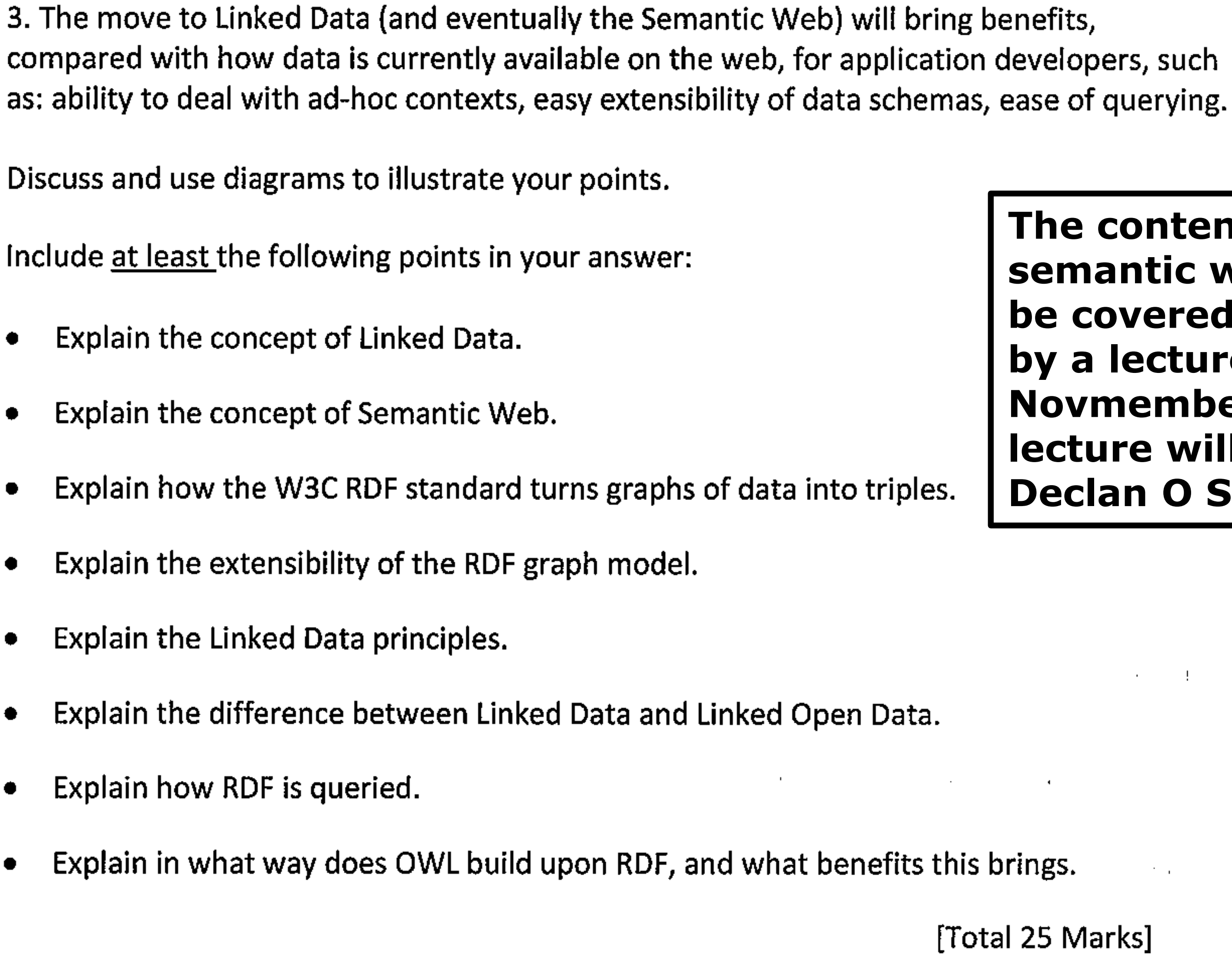
IV. Return only the first of the firstnames for each person in the document.

[13 Marks]

[Total 25 Marks]

2017

Past Semantic Web Exam Question



3. The move to Linked Data (and eventually the Semantic Web) will bring benefits, compared with how data is currently available on the web, for application developers, such as: ability to deal with ad-hoc contexts, easy extensibility of data schemas, ease of querying.

Discuss and use diagrams to illustrate your points.

Include at least the following points in your answer:

- Explain the concept of Linked Data.
- Explain the concept of Semantic Web.
- Explain how the W3C RDF standard turns graphs of data into triples.
- Explain the extensibility of the RDF graph model.
- Explain the Linked Data principles.
- Explain the difference between Linked Data and Linked Open Data.
- Explain how RDF is queried.
- Explain in what way does OWL build upon RDF, and what benefits this brings.

[Total 25 Marks]

The content for the semantic web question will be covered in and directed by a lecture on Monday Novmember 26th. The lecture will be given by Declan O Sullivan

2017

XQuery

- F,L,W,O,R Clauses
- Other Clauses- Slides included in today’s set.
  - Sequence: Union, Intersect, Accept
  - Conditional: If Then Else
  - Quantified Expression Clauses:Some, Every
- Built in Functions (e.g. string(), true(), not() etc.)
- Operators (e.g. - + != etc.)
- Use Defined functions- Slides covered in today’s class

User Functions

- XQuery allows users to define functions of their own
  - A function may take zero or more parameters.

  - A function definition must specify the name of the function and the names of its parameters if they exist
    - It may optionally specify types for the parameters
      - If no type is specified for a function parameter, that parameter accepts values of any type.
    - It may optionally specify types for the result of the function.
      - If no type is specified for the result of the function, the function may return a value of any type.

  - Body of the function is an expression enclosed in curly braces.

Example Function clause

XML Source
Tcd.xml

```
<?xml version="1.0"?>
<assessments>
  <student name="Smith">
      <mark thecourse="4BA5"> 99
         </mark>
      <mark thecourse="4BA1"> 75
         </mark>
  </student>
  <course name="4BA1"
takenby="Smith,Jones">
     <mark>60</mark>
  </course>
  <course name="4BA5"
             takenby="Smith,Bond">
     <mark>70</mark>
  </course>
</assessments>
```

XML Source
tcd2002.xml

```
<?xml version="1.0"?>
<assessments>
  <student name="Ledwidge">
      <mark thecourse="4BA5"> 45
</mark>
      <mark thecourse="4BA1"> 55
</mark>
  </student>
  <student name="ONeill">
      <mark thecourse="4BA5"> 85
</mark>  </student> </assessments>
```

XQuery

```
declare function local:all_students()
   {
for $s in
doc("data/tcd.xml")/assessments/student
union
doc("data/tcd2.xml")/assessments/studen
t
return
<student>
         {$s/@name}
         {$s/mark/@thecourse}
</student>
   };

<all>
{local:all_students()}
</all>
```

Result

```
<all>
    <student name="Smith" thecourse="4BA5"
thecourse="4BA1"/>
    <student name="Ledwidge" thecourse="4BA5"
thecourse="4BA1"/>
    <student name="ONeill" thecourse="4BA5"/>
</all>
```

Example Function clause with param

XML Source
Tcd.xml

```
<?xml version="1.0"?>
<assessments>
  <student name="Smith">
      <mark thecourse="4BA5"> 99
        </mark>
      <mark thecourse="4BA1"> 75
        </mark>
  </student>
  <course name="4BA1"
takenby="Smith,Jones">
     <mark>60</mark>
  </course>
  <course name="4BA5"
             takenby="Smith,Bond">
     <mark>70</mark>
  </course>
</assessments>
```

XML Source
tcd2.xml

```
<?xml version="1.0"?>
<assessments>
  <student name="Ledwidge">
      <mark thecourse="4BA5"> 45
</mark>
      <mark thecourse="4BA1"> 55
</mark>
  </student>
  <student name="ONeill">
      <mark thecourse="4BA5"> 85
</mark>  </student> </assessments>
```

XQuery

```
declare function local:
find_students($stuname as xs:string)
   {
for $s in
doc("data/tcd.xml")/assessments/student
union
doc("data/tcd2.xml")/assessments/studen
t
where $stuname = string($s/@name)
return
<student>
         {$s/@name}
         {$s/mark/@thecourse}
</student>
   };

local:find_students("Smith")
```

Result

```
<student name="Smith" thecourse="4BA5"
thecourse="4BA1"/>
```

Example Function clause with output type declared

XML Source
Tcd.xml

```
<?xml version="1.0"?>
<assessments>
  <student name="Smith">
        <mark thecourse="4BA5"> 99
           </mark>
        <mark thecourse="4BA1"> 75
           </mark>
  </student>
  <course name="4BA1"
takenby="Smith,Jones">
      <mark>60</mark>
  </course>
  <course name="4BA5"
            takenby="Smith,Bond">
      <mark>70</mark>
  </course>
</assessments>
```

XML Source
tcd2.xml

```
<?xml version="1.0"?>
<assessments>
  <student name="Ledwidge">
        <mark thecourse="4BA5"> 45
</mark>
        <mark thecourse="4BA1"> 55
</mark>
  </student>
  <student name="ONeill">
        <mark thecourse="4BA5"> 85
</mark>  </student> </assessments>
```

XQuery

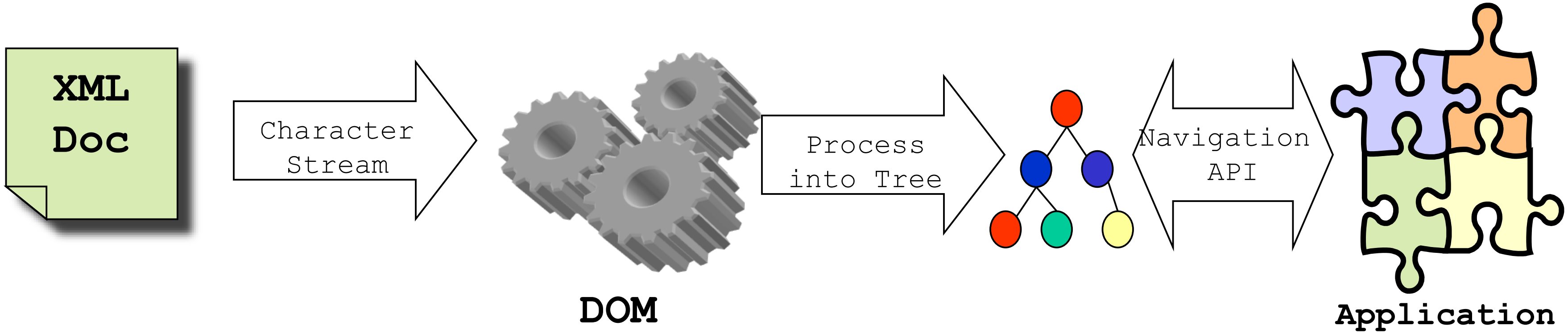
```
declare function local:all_students()
as element()*
   {
for $s in
doc("data/tcd.xml")/assessments/student
union
doc("data/tcd2.xml")/assessments/studen
t
return
<student>
         {$s/@name}
         {$s/mark/@thecourse}
</student>
   };

<all>
{local:all_students()}
</all>
```

Result

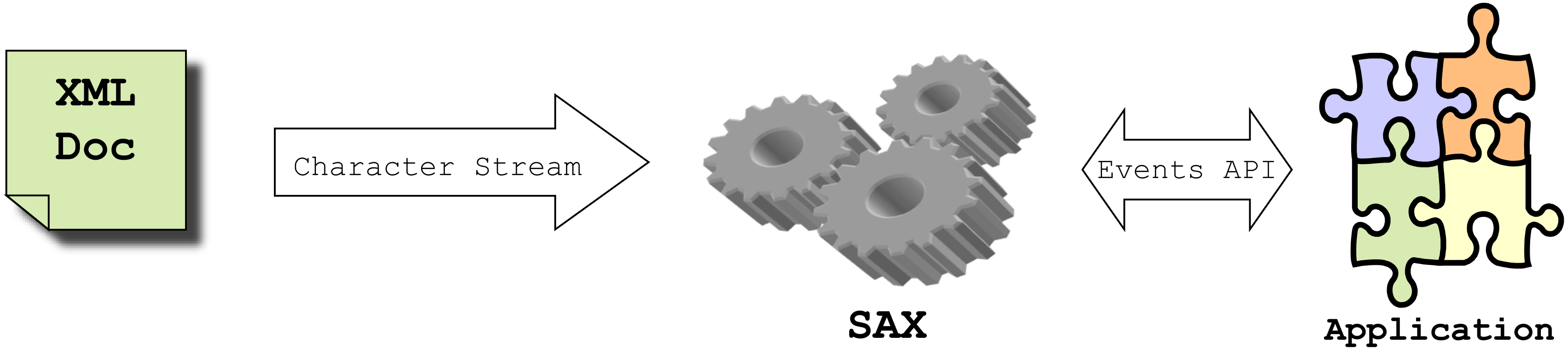
```
<all>
    <student name="Smith" thecourse="4BA5"
thecourse="4BA1"/>
    <student name="Ledwidge" thecourse="4BA5"
thecourse="4BA1"/>
    <student name="ONeill" thecourse="4BA5"/>
</all>
```

XML Processing: DOM Processing



- DOM stands for Document Object Model
- It views an XML tree as a data structure
- It is quite large and complex...
  - Level 1 Core: W3C Recommendation, October 1998
    - primitive navigation and manipulation of XML trees
    - other Level 1 parts: HTML
  - Level 2 Core: W3C Recommendation, November 2000
    - adds Namespace support and minor new features
    - other Level 2 parts: Events, Views, Style, Traversal and Range
  - Level 3 Core: W3C Working Draft, April 2002
    - adds minor new features
    - other Level 3 parts: Schemas, XPath, Load/Save

XML Processing: SAX Processing



- SAX stands for Simple API for XML
- An XML tree is not viewed as a data structure, but as a stream of events generated by the parser.
- The kinds of events are:
  - the start of the document is encountered
  - the end of the document is encountered
  - the start tag of an element is encountered
  - the end tag of an element is encountered
  - character data is encountered
  - a processing instruction is encountered
- Scanning the XML file from start to end, each event invokes a corresponding callback method that the programmer writes

Programmatic APIs

- Different programming APIs to connect to BaseX XML database
  - REST-Style Web API
  - Variety of Client APIs for different programming languages
- See http://docs.basex.org/wiki/Developing

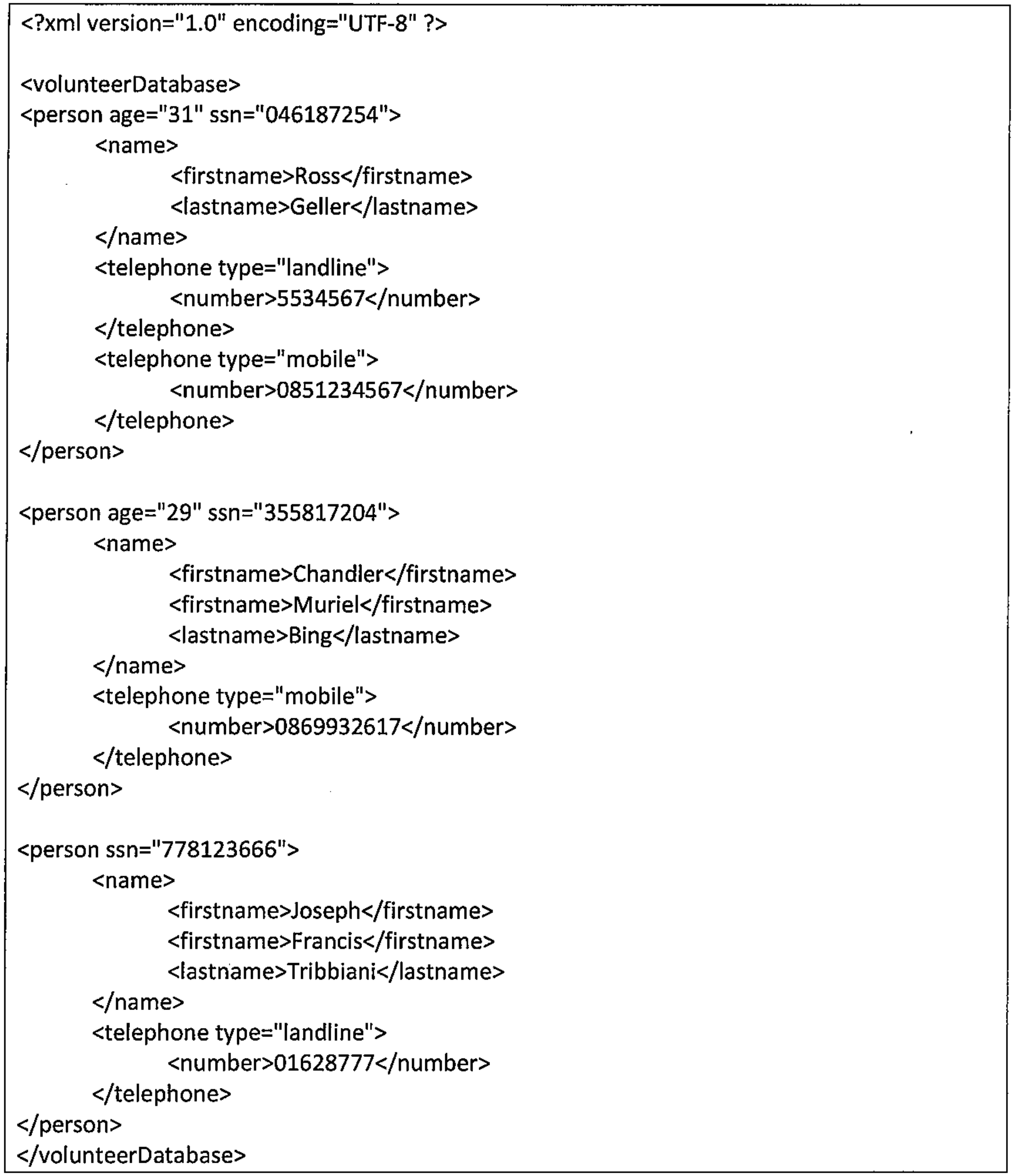
Exercises on Blackboard

1. DTD
2. Well Formed XML
3. Xpath
4. Xquery

- View exercises from Blackboard and choose an exercise to do during this class
- Work on it in groups of 2 or 3
- Bring your answers to the lab sessions if you want feedback.

Exercise 1 **Suggest a DTD for the XML file in Figure A**

FIGURE A:



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

<volunteerDatabase>
<person age="31" ssn="046187254">
        <name>
                <firstname>Ross</firstname>
                <lastname>Geller</lastname>
        </name>
        <telephone type="landline">
                <number>5534567</number>
        </telephone>
        <telephone type="mobile">
                <number>0851234567</number>
        </telephone>
</person>

<person age="29" ssn="355817204">
        <name>
                <firstname>Chandler</firstname>
                <firstname>Muriel</firstname>
                <lastname>Bing</lastname>
        </name>
        <telephone type="mobile">
                <number>0869932617</number>
        </telephone>
</person>

<person ssn="778123666">
        <name>
                <firstname>Joseph</firstname>
                <firstname>Francis</firstname>
                <lastname>Tribbiani</lastname>
        </name>
        <telephone type="landline">
                <number>01628777</number>
        </telephone>
</person>
</volunteerDatabase>
```

EXAMPLE DTD to show SYNTAX

```
<!DOCTYPE NEWSPAPER [

<!ELEMENT NEWSPAPER (ARTICLE+)>
<!ELEMENT ARTICLE
      (HEADLINE,BYLINE+,LEAD?,BODY,NOTES*)>
<!ELEMENT HEADLINE (#PCDATA)>
<!ELEMENT BYLINE (#PCDATA)>
<!ELEMENT LEAD (#PCDATA)>
<!ELEMENT BODY (#PCDATA)>
<!ELEMENT NOTES (#PCDATA)>

<!ATTLIST ARTICLE AUTHOR CDATA #REQUIRED>
<!ATTLIST ARTICLE EDITOR CDATA #IMPLIED>
<!ATTLIST ARTICLE DATE CDATA #IMPLIED>
<!ATTLIST ARTICLE EDITION CDATA #IMPLIED>

<!ENTITY NEWSPAPER ”Trinity Times">
<!ENTITY PUBLISHER ”Trinity Press">
<!ENTITY COPYRIGHT "Copyright 1998 TCD Press">

]>
```

- Syntax for occurrences of elements in DTDs

  ?: zero-or-one

  +: one-or-more

  *: zero-or-more

Exercise 2:Design XPath queries for

1. Get all the firstnames of persons in the file (without using //)

2. Get just the text from the firstname elements of name

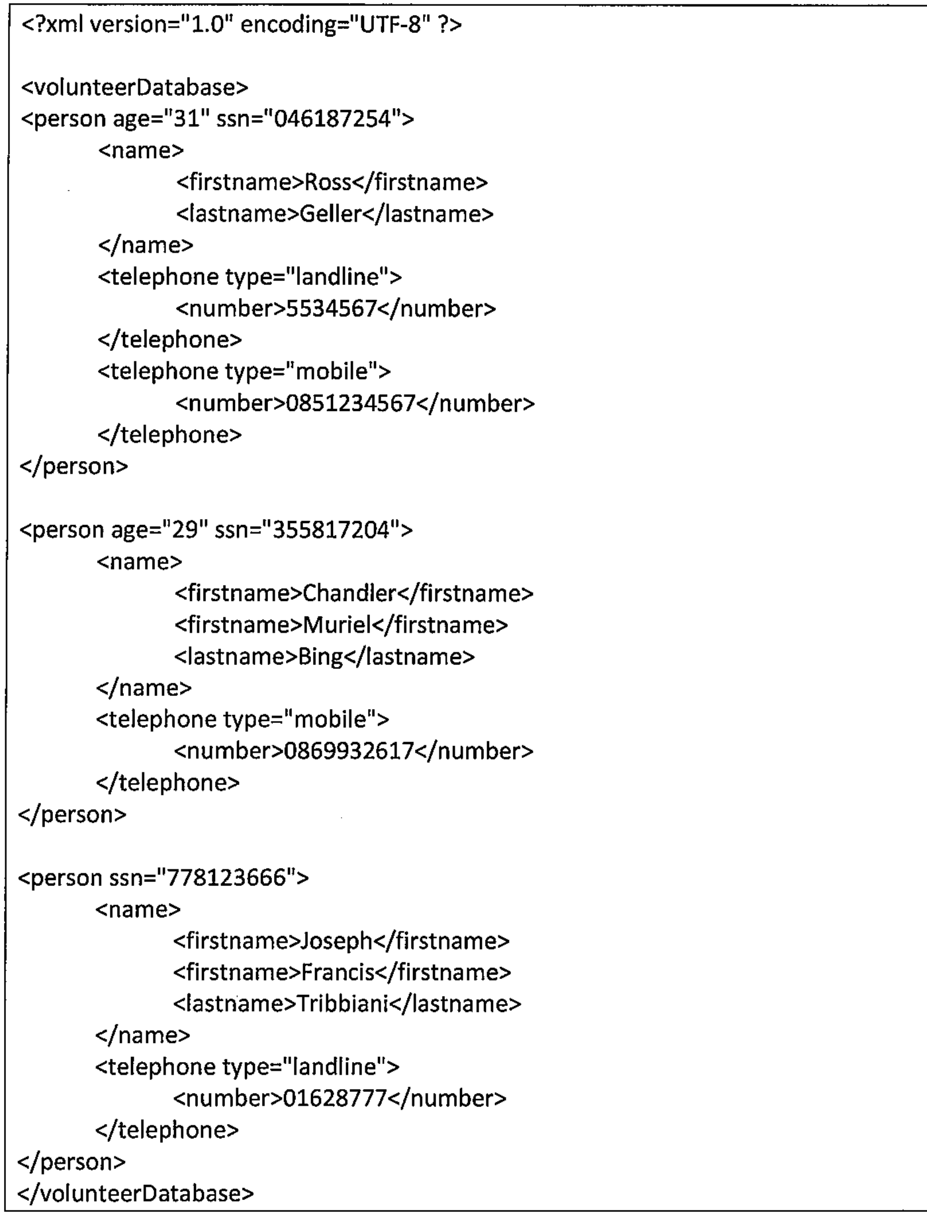
3. Return only the person elements that has an age attribute

4. Return the entire person element whose firstname is "Chandler"

5. Return all the values of ssn attributes

6. Return the number of each person who has an age attribute

FIGURE A:



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

<volunteerDatabase>
<person age="31" ssn="046187254">
        <name>
                <firstname>Ross</firstname>
                <lastname>Geller</lastname>
        </name>
        <telephone type="landline">
                <number>5534567</number>
        </telephone>
        <telephone type="mobile">
                <number>0851234567</number>
        </telephone>
</person>

<person age="29" ssn="355817204">
        <name>
                <firstname>Chandler</firstname>
                <firstname>Muriel</firstname>
                <lastname>Bing</lastname>
        </name>
        <telephone type="mobile">
                <number>0869932617</number>
        </telephone>
</person>

<person ssn="778123666">
        <name>
                <firstname>Joseph</firstname>
                <firstname>Francis</firstname>
                <lastname>Tribbiani</lastname>
        </name>
        <telephone type="landline">
                <number>01628777</number>
        </telephone>
</person>
</volunteerDatabase>
```

Exercise 3

Explain using examples what constitutes a well formed and valid XML document.

- **Well formed-** XML Declaration required,
- Exactly one root element,
- Empty elements are written in one of two ways:Closing tag or Special start tag, For non-empty elements, closing tags are required,
- Attribute values must always be quoted,
- Start tag must match closing tag (name & case),
- Correct nesting of elements

Exercise 4

- NOW using **bib.xml (on next slide) List books published by Addison-Wesley after 1991, including their year and title.**

You should get

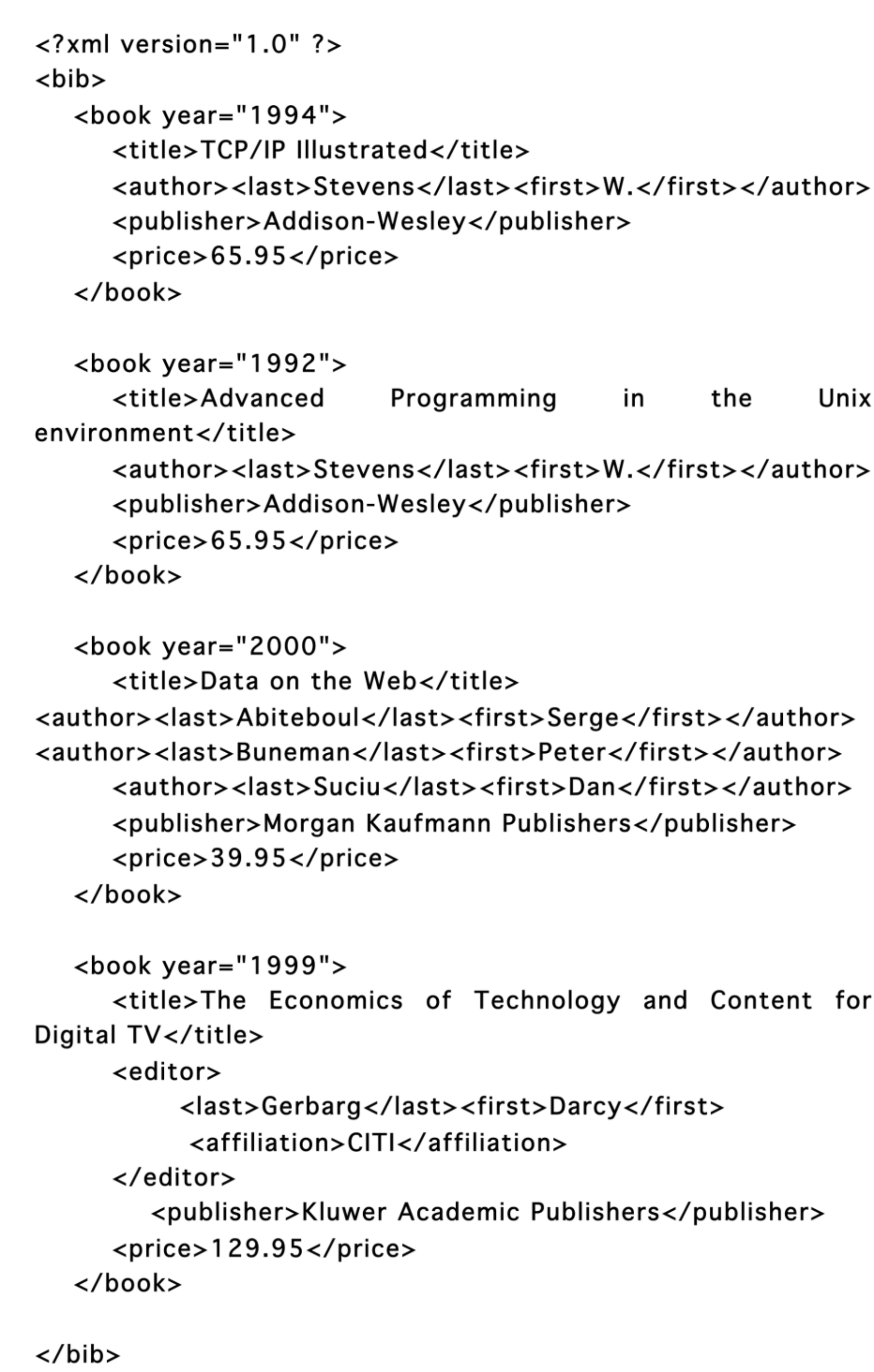
```
<bib>
   <book year="1994">
      <title>TCP/IP Illustrated</title>
   </book>
   <book year="1992">
      <title>Advanced Programming in the Unix environment</title>
   </book>
</bib>
```

- Example syntax

```
let $c:=
   doc("data/tcd.xml")/assessments
   /course/mark
return
   <list_of_avg_course_marks>
   {$c}
   </list_of_avg_course_marks>

for $j in
        doc("data/tcd.xml")/assessments/course/@name
return
   <one_of_courses_is>
   {$j}
   </one_of_courses_is>
```

Bib.xml



```
<?xml version="1.0" ?>
<bib>
   <book year="1994">
      <title>TCP/IP Illustrated</title>
      <author><last>Stevens</last><first>W.</first></author>
      <publisher>Addison-Wesley</publisher>
      <price>65.95</price>
   </book>

   <book year="1992">
      <title>Advanced     Programming     in     the     Unix
environment</title>
      <author><last>Stevens</last><first>W.</first></author>
      <publisher>Addison-Wesley</publisher>
      <price>65.95</price>
   </book>

   <book year="2000">
      <title>Data on the Web</title>
<author><last>Abiteboul</last><first>Serge</first></author>
<author><last>Buneman</last><first>Peter</first></author>
      <author><last>Suciu</last><first>Dan</first></author>
      <publisher>Morgan Kaufmann Publishers</publisher>
      <price>39.95</price>
   </book>

   <book year="1999">
      <title>The  Economics  of  Technology  and  Content  for
Digital TV</title>
      <editor>
           <last>Gerbarg</last><first>Darcy</first>
            <affiliation>CITI</affiliation>
      </editor>
         <publisher>Kluwer Academic Publishers</publisher>
      <price>129.95</price>
   </book>

</bib>
```

XPath and XQuery Labs Dates

- Monday 12th November 2018 10 to 11am
  - Groups 16 to 22 inclusive
- Monday 12th November 2018 11am to 12 noon
  - Groups 8 to 15 inclusive
- Thursday 15th November 2018 11am to 12 noon
  - Groups 1 to 7 inclusive
- Venue: ICT 1 Lab Assistant Ademar Crotti

Demos Dates

- Monday 19th November 2018 10 to 11am
  - Groups 16 to 22 inclusive

- Monday 19th November 2018 11am to 12 noon
  - Groups 8 to 15 inclusive

- Thursday 22nd November 2018 11am to 12 noon
  - Groups 1 to 7 inclusive

- Venue: ICT 1 Lab Assistant Ademar Crotti