3. XPath with Identities and First Steps with XSLT

The purpose of this lab is to practice

1. using identities and identity references in XPath
2. writing simple XSLT stylesheets to transform XML documents.

3.1 XPath with IDs and IDREFs

Open the `courses-ID.xml` file in Eclipse.[1] Write XPath expressions that return answers to queries below.[2] The queries train, among other things, the use of IDs and IDREFs.

1. Which course has the lowest enrollment? Note: There are some courses without enrollment figures.
2. Which courses have at least two instructors?
3. Return the titles of the courses taught by Prof. Aiken.
4. Who are the co-teachers of Jerry Cain, that is, which people are jointly teaching a course with Jerry Cain?
5. Which are the courses taught by Jerry Cain, but not by Mehran Sahami?
6. Return the titles of courses taught by instructors from the department of Linguistics.
7. Which courses have prerequisite courses that are taught at another department than the course itself?
8. Which courses are taught by an instructor who also teaches one of the courses prerequisites?
9. Which courses have a lower enrollment than any of their prerequisites?
10. Which courses are taught by a lecturer?
11. Which instructors (i.e., professors or lecturers) do not have a middle initial?

[1] The data are taken, with slight modification of the ID's, from the website of the online course "Introduction to Databases" by Jennifer Widom at `http://openclassroom.stanford.edu`.

[2] The queries are original.

3.2 First Steps with XSLT

Saxon is the most widely used Java-based XSLT and XQuery processor. There exist three editions of Saxon, one of which, the Home Edidition, is free.
In the lab exercises, we will Kernow, a graphical frontend for Saxon. You can download Kernow from

`http://kernowforsaxon.sourceforge.net/`

Kernow is a Java package and can be run under Linux, Windows and Mac OS.
You can develop your stylesheets in the XSLT Sandbox of Kernow. The outermost element of your stylesheets should be of the kind `xsl:stylesheet`, with attributes set as shown below:

```
<xsl:stylesheet version="2.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

In the lectures, we have introduced XSLT 1.0, because version 2.0 is not yet widely supported. Saxon, however, is a processor for XSLT 2.0 and will return a warning if you declare your stylesheet as of version 1.0.
To tell Kernow that the output should be XML 1.0 and that we would like it properly indented, add as the first element in the stylesheet the following output specification:

```
<xsl:output method="xml"
            version="1.0"
            encoding="iso-8859-1"
            indent="yes"/>
```

The version of Kernow that you download allows you to run 100 queries. Then it will ask you to enter a code in order to proceed. I have the code and will send it to you if you ask me by email.
These exercises are largely similar to the ones for XQuery. This is intentional, since both XQuery and XSLT allow one to specify transformations turning XML documents into other XML or (X)HTML documents.
The exercises allow you to compare the means by which such transformations can be expressed in the two languages.
The version of Kernow that you download allows for 100 executions of stylesheets. Then it will ask you to enter a code in order to proceed. I have the code and will send it to you if you ask me by email.

Recipe Transformations

The following transformations are over the `recipes.xml` document. Formulate your transformations in such a way that they return the correct result for all possible recipe documents that satisfy `recipes.dtd`.

1. Return a list, inside an element <recipes>, of recipes, containing for every recipe the recipe's title element and an element with the number of calories. Use different approaches:

   (a) express iteration by recursive calls of templates

(b) express iteration by <xsl:for-each> elements.

Create new elements

(a) by explicit construction, that is by writing the tags into the code,

(b) by dynamic construction, that is, by using <xsl:element> and <xsl:attribute> elements,

(c) by shallow and deep copying, wherever the latter is possible.

2. Using iteration by recursion, return a similar list, alphabetically ordered according to title.

3. Using iteration by means of <xsl:for-each>, return a similar list, ordered according to calories in descending order.

   (**Hint:** Consult the Web, e.g. the site www.w3schools.com, if you encounter difficulties with the ordering.)

4. Return a similar list, with title as attribute and calories as content.

5. Return a list, inside an element <recipes>, of recipes, where each recipe contains the title and the top level ingredients, while dropping the lower level ingredients.