2. Unit: XSLT

Exercise 2.1 (Recursion in Data)

(a) Write an XSLT stylesheet which maps the structures of the seas and rivers from Mondial in the following way: Every sea element should contain the name of the sea and a river element for each river flowing into that sea. Each river element, again, should recursively contain a river/lake element for each river/lake flowing into it, and so on, as sketched below. For rivers, their tributaries should be ordered according to the elevation if their estuaries (unknown ones last). Put the lakes in-between accordingly.

```
<waters>
  <sea>
    <name>North Sea</name>
     :
    <river>
      <name>Rhein</name>
      <length>...</length>
      <river>
        <name>Main</name>
        <length>...</length>
        <river>
          <name>Tauber</name>
          <length>...</length>
        </river>
        :
      </river>
      <river>
        <name>Neckar</name>
        <length>...</length>
        <river>
      </river>
      :
      <lake>
        <name>Bodensee</name>
          :
      </lake>
    </river>
  </sea>
</waters>
```

(b) Write another stylesheet (that uses the output of the above one as input) which computes for each river that flows into a sea the total sum of the length of all rivers flowing (directly or transitively) into it, and output the results into a table.

(c) Write another stylesheet (that uses the original mondial.xml! as input) which computes for each river that flows into a sea the total sum of the length of all rivers flowing (directly or transitively) into it, and output the results into a table.

(d) Extend the stylesheet from part (a) such that it allows to answer the following things by XPath queries:

(i) Is it possible that rain falls south of the equator, and the water then flows into the Mediterranean Sea?

(ii) From which countries there is water flowing into the Black Sea?

In contrast to XQuery, XSLT allows to generate *multiple output files* in a single run. This is e.g. required when generating (static or dynamic) Web pages from a database.

If you put a directory with name `public_html` into your home directory in your IFI pool account, is will be available on the Web server under the URL `http://user.informatik.uni-goettingen.de/~`*username*`/`. (check this by putting a simple index.html there). Create and use a `mondial` directory there for this exercise (then you can remove it completely with cd'ing to `public_html` and executing `rm -rf mondial`).

The next two exercises are strongly intertwined. If they are solved by several people in the group, discuss before what agreement is necessary such that both of them fit together.

Exercise 2.2 (Generation of Web Pages (1: static Web pages))

Create the following directories and static HTML files by an XSLT stylesheet (in case that directory names or filenames would contain spaces, use a "_" instead):

- for each country, a directory named with the car code of the country.
- inside this, an `index.html` that contains some information about the country (you can also add the flag from its wikipedia page, and a reference to the wikipedia page – guessing the wikipedia page URL is a case where exception handling can be used if the guessed URL is not the correct one),
- inside the country directory, a directory for each of its provinces (if the country has provinces), again with an `index.html` with information about the province.
- inside the country or province directories, a directory for each of the cities, again with an `index.html`.
- the country's `index.html` should have a reference to the capital city of the country, and references to the other cities. Each city should also have a reference of the form "x is located in the y province of z" to the country and the province.

(For a smaller XML sample, take `mondial-europe.xml` from the Mondial Web page)

Exercise 2.3 (Generation of Web Pages (2: dynamic Web pages))

Dynamically generated Web pages consist of relevant XML data associated with an XSL stylesheet that is applied upon access to that XML file.

In this exercise, two XSL stylesheets should be written. The first one does the following (in case that filenames would contain spaces, use a "_" instead):

- for each organization, create a (static) XML file *`abbrev`*`.xml` (e.g., `public_html/mondial/EU.xml`) which contains the relevant information related to the organization.

These generated XML files should at its beginning contain a reference to another XSLT stylesheet (which is again generic for all organizations) which dynamically creates an HTML Web page about the organization as follows:

- a short text with the name and abbreviation of the organization,
- (if its establishment date is known) a short statement how old (in days) the organization is in the moment when the page is accessed!
  (that's the main reason why here a dynamically created HTML page is used)
- including the list of all members with references to them (i.e., the country pages generated in the previous exercise), and a reference to the city (again, to the page generated in the previous exercise) where the headquarter is located.