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 of 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>Mosel</name>
        <length>...</length>
        <river>
          <name>Saar</name>
          <length>...</length>
        </river>
        :
      </river>
      <river>
        <name>Neckar</name>
        <length>...</length>
        <river>
      </river>
      :
      <lake>
        <name>Bodensee</name>
        <river>
          <name>Bregenzer Ach</name>
          <length>...</length>
        </river>
          :
      </lake>
    </river>
  </sea>
</waters>
```

(b) What waters are still missing?
(if you have no idea, state an appropriate query against both versions and find out what the missing waters have in common.)
Add them appropriately.

(c) 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 length of its river system (i.e., itself, and all rivers flowing directly or indirectly into it), and output the results into a table.

(d) 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 indirectly) into it, and output the results into an HTML table.

(e) Extend the stylesheet from part (a) such that its output contains enough information to answer

the following things by XPath queries (and give the 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?

(iii) is there a province-level entity in Europe from which water flows into each of the following seas: Mediterranean Sea, Black Sea, North Sea?

Exercise 2.2 (Generation of Web Pages)

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, it 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 (it can be removed completely with `cd ~/public_html` and executing `rm -rf mondial`).

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):

a) a directory `countries/`, and inside this

b) for each country, a directory named with the car code of the country.

c) intermediate step: 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).
(to view the result, the browser must call the URL `http://user.informatik.uni-goettingen.de/~`*username*`/countries/`*code*`.xml`.

d) 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.

e) inside the country or province directories, a directory for each of the cities, again with an `index.html`.

f) the country's `index.html` should have a reference to the page of the capital city of the country, and clickable references to the pages of the other cities. Each city should also have a clickable reference of the form "x is located in the y province of z" to the HTML pages of the country and the of the province.

g) an alternative to the countries' index.html files by *dynamic Web pages*:
- for each country, create a (static) XML file directly in the mondial directory *`code`*`.xml` (e.g., `public_html/mondial/D.xml`) which contains the relevant information related to the country.
- Each of these generated XML files should at its beginning contain a reference to another XSLT stylesheet (which is again generic for all countries) which *dynamically* creates an HTML Web page about the country as above, but where the *current population* is computed by using the most recent available population count and the *population growth rate*.
(To view the result, the browser must call the URL `http://user.informatik.uni-goettingen.de/~`*username*`/countries/`*code*`.xml`).
NOTE: this calculation might not work in all browsers because the math:pow function is not supported

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

Exercise 2.3 (XML to LaTeX)

Write an XSLT stylesheet that creates a LaTeX file `hamlet.tex` from `hamlet.xml`:

- Act headers and Scene headers as section/subsections,
- Act and Scene intros, and stage directives as quote environments,
- Speeches as items in a description environment, with the name of the person as item label,
- XSL programming: for all mentions of country names and city names that exist in Mondial, print these names in red,
- Perfection (requires knowledge of Linked Open Data, LaTeX, and maybe even system administration): for all of them, print the URL of it in the Mondial LOD server (`https://www.semwebtech.org/mondial/10`), and make the link clickable in the generated pdf. (For viewing, use a browser, not a PDF viewer, to use the clickable link).