4. Unit: XSLT

Exercise 4.1 (XML to HTML) Write an XSLT routine performing the following task:
Map the following country data for each country to an HTML table:

- country name
- car code
- capital's name
- number of inhabitants
- the names of all listed cities, inside a nested html table.

Exercise 4.2 (Arithmetic Terms) Arithmetic terms over integer values and operators $+, -, *$ and *div* (integer division) can be represented by their syntax trees, with the syntax trees given in XML. A possible XML notation for syntax trees is given in the following example for the term

$$4 + ((7 - 2)\ div\ 2)$$

```
<term>
    <plus>
        <val>4</val>
        <div>
            <minus>
                <val>7</val>
                <val>2</val>
            </minus>
            <val>2</val>
        </div>
    </plus>
</term>
```

a) Write down the syntax tree for the term $((91\ div\ (19 - (3 * 8))) + 3)$, using the XML notation from the above example.

b) Write a DTD for the given notation. Each term should be considered a single XML document instance.

c) Write three XSLT stylesheets that take a syntax tree in the notation depicted above as input, and produce as output
   - the term as text in *inorder* notation (outcome should be `(4+((7-2) div 2))` for the example),
   - the term as text in *preorder* notation (outcome should be `+ 4 div - 7 2 2`), and
   - the term as text in *postorder* notation (outcome should be `4 7 2 - 2 div +`).

   Test the stylesheets using the term $((91\ div\ (19 - (3 * 8))) + 3)$ as input.

d) Write an XSLT stylesheet that evaluates a syntax tree in the notation depicted above.

Exercise 4.3 (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 must contain the name of the sea and a river element for each river flowing into that sea. Each river element, again, must recursively contain a river element for each river flowing into it, and so on:

```
<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>
      :
    </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.