2. Unit: XSLT

Exercise 2.1 (XML to HTML with XSLT)

Solve Exercise 1.5 in XSLT.

Exercise 2.2 (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.

Exercise 2.3 (Grouping an Input Sequence) Write an XSLT program that gets an input sequence of simple elements of the form given below (this file is linked as "DuplicateInput.xml" on the Web page): The final result should be the following: a table with columns called d,a,c,b. For the i-th table row, the element of the respective column should be the i-th element of that type in the input sequence, i.e.,

d	\mathbf{a}	$^{\mathrm{c}}$	b
d: 31	a: 2	c: 43	b: 47
d: 42	a: 8	c: 65	b: 43
:	:	:	:

Preferably, produce the output as an HTML table.

This exercise illustrates a lot of features and properties of programming language concepts. The task seems to be quite obvious and easy. By hand, writing the result on a paper, this can be done quite easily by traversing the input sequence once. It is recommended to proceed as follows:

- a) figure out how to do it in an imperative language, e.g., Java or in detailed pseudocode (decide which data structures to use and how to maintain them),
- b) write an XSLT stylesheet that just produces the table header.
- c) assume that you already know that the columns are a,b,c,d and fill them as above.
- c) combine both parts by holding b) in a variable and using it as a base for c).

Think about alternative ways to solve it, and give a complexity estimation.

Exercise 2.4 (XSLT and Recursive Data Structures: Binary Search Trees)

- a) Create a sample instance for XML Markup of *Binary Search Trees* (probably use the Abstract Datatype definition from the CS I lecture). Note: also an empty BST is a BST.
- b) give a DTD-style description (note that since a BST can either be empty or not, the expressiveness of DTDs is not adequate). Optionally: give an XML Schema description.
- c) write XSL stylesheets for (i) inserting elements into the tree, (ii) outputting the tree contents in inorder, (iii) outputting the tree in a graphical way (e.g., nested tables) in HTML, (iv) searching, and (iii) deleting elements in the tree.

(e.g., invoked by saxonXSL -o mybsb.xml mybsb.xml bsb-insert.xsl insert=6)

Exercise 2.5 (Stylesheet for Schedules)

Write an XSL stylesheet for transforming schedule data given in the following form into an html calendar table:

```
<?xml version="1.0" encoding="UTF-8"?>
<schedule name="Meine Termine">
   <year n="2006">
        <month n="1">
            <day n="3">
                <date starttime="15:00">
                    <name>Institutsbesprechung</name>
                </date>
            </day>
        </month>
        <month n="2">
            <day n="2">
                <date starttime="18:00">
                    <name>Telefonat</name>
                </date>
            </day>
        </month>
    </year>
</schedule>
```

The resulting schedule should have about the following layout:

jan	feb	mar	apr	may	jun	jul	aug	sep	oct	nov	dec
1	1	1	1	1	1	1	1	1	1	1	1
2	2 18:00 - Telefonat	2	2	2	2	2	2	2	2	2	2
3 15:00 - Institutsbesprechung	3	3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4	4	4
:	:	:	:	:	:	:	:	:	:	:	: