XML Technologies: Exercise 4

XSLT

This is a mandatory exercise and the result will be part of your final mark. The solution must be uploaded to OLAT by **April 15th at 15:59**. Late submissions will not be accepted.

Submit the following files in a zipped archive:

- tablify.xsl
- euler.xsl

Make sure the archive is named [lastname]_[firstname]_4.zip (for example *mueller_mathias_4.zip*. Some parts of your submission may be automatically evaluated, so make sure to name your files *precisely* as prescribed, otherwise you might not get any points.

1 Whisky in the Browser (2 points)

XSLT is a declarative programming language. Its purpose is to transform XML documents – either to another XML document or to any other text-based format. One of the most widespread uses of XSLT is transforming XML into XHTML – in order to display the result in the web.

The XML document `whiskies.xml` is fine as as format to store or exchange data. However, XML syntax has nothing to say on the subject of *presentation*, that is, displaying the data in an accessible manner. If all our data about whiskies were to be translated to HTML instead, this is what the result could look like in all major browsers:

ID	distillery	bottling	alcohol by volume	age
1	Kilchoman	Machir Bay	46%	6
2	Ardbeg	Corryvreckan	57.1%	NAS
3	Bruichladdich	The Laddie Ten	50.0%	10

Write a complete XSLT stylesheet that transforms the original `whiskies.xml` document to XHTML and displays the data in a table. When opened in your browser, the page should look like the table above.

Be sure not to hard-code any table cell values but actually retrieve them from the input XML document. For instance, an additional `whisky` element should not cause your stylesheet any trouble.

Save your stylesheet as `tablify.xsl`. If you have `xsltproc` installed, you can test your submission as follows:

```
xsltproc tablify.xsl whiskies.xml > table.html
```

2 Recursive XSLT: First Euler Problem (3 points)

The XSLT stylesheet you wrote for question **??** is pretty short and straightforward. Yet, many real-world applications take this to quite another level of sophistication. Applications of more advanced XSLT code are, for instance,

- organizing XSLT code in separate modules that import or include one another, with an explicit hierarchy (modularisation)
- multi-step transformations with different template modes, in order to reiterate XML data that was processed already (where a template was applied already)
- recursive named templates that search arbitrarily deep structures or perform any other task an arbitrary number of times, depending on an input parameter

In this question, you will write a recursive template. Consider the following problem:

If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23. Find the sum of all the multiples of 3 or 5 below 1000. [1]

Suppose there already was an XSLT solution – but somehow, the named template disappeared:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
    <xsl:output indent="no" omit-xml-declaration="yes"/>

    <xsl:template match="/">
```

[1] The problem description is also available on the Euler project page: `https://projecteuler.net/problem=1`.

```
        <!--$k stores the value up to which multiples should be summed.-->
        <xsl:variable name="k" select="1000"/>

        <!--$multiples stores the result of invoking the named template.-->
        <xsl:variable name="sum">
            <xsl:call-template name="sum-multiples">
                <xsl:with-param name="k" select="$k"/>
            </xsl:call-template>
        </xsl:variable>

        <!--Writing the final result to an output document.-->
        <xsl:value-of select="$sum"/>
    </xsl:template>

    <xsl:template name="sum-multiples">
        <!--Your code here-->
    </xsl:template>

</xsl:transform>
```

The incomplete stylesheet above is also available as `euler.xsl`. The input to this stylesheet is irrelevant, but transform `euler.xml`, which you will also find attached.

Complete the stylesheet above by writing a named template that

- **integrates well with the scaffold that is already given (i.e. that does not require any change to the code outside of the named template)**
- **takes an input parameter called "k" where a number can be specified up to which multiples are found**
- **returns a single number (that is the sum of all numbers between 1 and `$k` that are multiples of 3 or 5) as an `xsl:sequence`. If you read out the value of `$sum` like this:**

  ```
  <xsl:value-of select="$sum"/>
  ```

  the result should be:

  ```
  233168
  ```

Save the complete stylesheet as `euler.xsl`.