Klausur "Semistructured Data and XML"
Summer Term 2021
Prof. Dr. Wolfgang May
17. August 2021, 11:00–13:00
Working Time: 120 Minutes
(carried out as a computer-based ILIAS exam)

Vorname:

Nachname:

Matrikelnummer:

Setting: The usage of saxon (with the aliases saxonValid, saxonXQ, and saxonXSL defined as in the course) and xmllint (validation error messages are better with xmllint) is recommended. Web access (e.g. for XPath/XQuery and XSLT documentation) is allowed. It was also recommended to have the slides, and a condensed self-prepared "cheat sheet" (preparation of a cheat sheet is a very effective way to work through the materials).
Answers might be given in English or German (most answers are program code anyway). In the text, the german translation is sometimes given in parentheses.
Give *all* answers via the ILIAS system.
Like in a "paper exam", also solutions that do maybe not work (or do not work completely) can be delivered and will be graded with appropriately partial points.
For **passing** the exam, **50** points are sufficient.

	Max. Punkte	Schätzung für "4"
Aufgabe 1 (XML)	20	15
Aufgabe 2 (DTD)	15	10
Aufgabe 3 (XPath (a))	2	2
Aufgabe 4 (XPath (b))	4	3
Aufgabe 5 (XPath/XQuery (c))	6	4
Aufgabe 6 (XPath/XQuery (d))	6	3
Aufgabe 7 (XPath/XQuery (e))	6	3
Aufgabe 8 (XPath/XQuery (f))	9	4
Aufgabe 9 (XPath/XQuery (g))	9	4
Aufgabe 10 (XSLT)	15	4
Aufgabe 11 (Miscellaneous (a))	4	4
Aufgabe 12 (Miscellaneous (b))	4	2
Summe	100	58

Note:

Project: All Years of Tour de France Database

All exercises are based on a common "project": a database about all "Tour de France" instances. The "Tour de France" is a cycling sports (and touristic) event that takes place every year usually during three weeks in the summer (except in 2020, when it was postponed to September) around France and sometimes also other (more or less) neighboring countries.

1. There is a *tour* instance in every year. As examples, we consider the 2020 and mainly the 2021 tour instances.

2. Every year, several *teams* participate; each with several *riders*.

3. So, for every year, it is stored which riders started for which teams. Riders may change teams from one year to the other.

4. For every rider that ever participated, the name, the birthdate, and the country of birth is stored (so we don't have to consider people changing nationalities).

Sample: In 2021, among others the *Jumbo Visma* team participated with riders *Primoz Roglic* (from *Slovenia*, born October 10th, 1989), *Wout van Aert* (from *Belgium*, born September 15th, 1994), *Sepp Kuss* (from the *USA*, born September 13th, 1994), *Jonas Vingegaard* (from *Denmark*, born December 10th, 1996) and others. Also, the *UAE Team Emirates* participated, with rider *Tadej Pogacar* (from *Slovenia*, born September 21st, 1998) and others. Team *"Trek"* participated with riders *Bauke Mollema* (from the *Netherlands*, born November 26th, 1986), *Kenny Elissonde* (from *France*, born July 22nd, 1991) and others.

In 2020, *Jumbo Visma* participated also with *Roglic*, *van Aert*, *Kuss* and some others. UAE also with *Pogacar* and *Vingegaard* (the latter was not true in reality, but serves here as an example for a rider who changed the team).

5. In every year, the tour consists of a sequence of *stages*; for each stage, the date is stored. Assume that there is at most one stage per day. Every stage is assigned a type: flat, hilly, mountains.

6. Every stage leads from one place to another. Places can be towns/cities, or mountains/passes. A subsequent stage does not necessarily start where the previous stage ended (but usually in the same region).

7. Places have a name and an elevation, and are located in a country (usually, but not always France).

8. For every stage, the starting place and the destination place, and optionally, a sequence of intermediate mountains/passes and places is stored.

Sample: In 2021, the first stage on June 26th led from *Brest* to *Landerneau*, it was a *hilly* stage of 198km. The 11th stage in 2021 on July 7th over 199km started in *Sorgues*, to crossed the *Mont Ventoux*, down to *Malaucene*, then to *Bedoin*, crossed the *Mont Ventoux* again, and finished in *Malaucene*. The 15th stage on July 11th lead over 191km through mountains from *Ceret* to the town *Andorra* (which is in the country *Andorra*); it crossed the *Port d'Envalira* and the *Col de Beixalis* (both also in the country *Andorra*). The (last) 21st stage on July 18th lead flatly over 108km from

Chatou to *Paris*.
In 2020, the first stage was a 156km hilly round trip from *Nizza* to *Nizza*, and the last, 21st stage was 122km flat from *Mantes* to *Paris*.

Brest, *Landerneau*, and *Nizza* are at an elevation of 10m, *Sorgues* is at 20m, *Chatou*, *Mantes*, and *Paris* are at 30m, *Ceret* is at 120m, *Malaucene* and *Bedoin* are at 300m, *Andorra* has 1011m, the *Mont Ventoux* has 1909m, the *Port d'Envalira* has 2407m, and the *Col de Beixalis* has 1795m.

9. For every stage, the result is stored: the order in which the riders *arrived* at the finish line together with the duration each of them needed for the stage.
Note that it is possible that riders abandon the tour at some day, then they will not be listed in remaining stages.

Sample: The above 11th (*Mont-Ventoux*-)stage 2021 was won by *Wout van Aert* in 5:17:43h, then followed *Elissonde* and *Mollema* (both 5:18:57), *Pogacar* and *Vingegaard* (5:19:21) (and later the rest).

The above 15th stage (to Andorra) was won by *Sepp Kuss* in 5:12:06h, and the others arrived later. For this example, it may be useful also to store that *Elissonde* arrived at 5:16:17h.

The 21st stage was also won by *Wout van Aert*, in 2:09:37h; Pogacar, Vingegaard and the others crossed the line with the same time directly after him.

Exercise 1 (XML [20 Points])
Design an XML structure (use the frame given in file `exam.xml`) and fill it with some sample data (e.g. with some of the example data given in the text).
(Information about handling dates and times in XML can be found in the course's slides around Slide 300.)

Copy-and-paste the XML from the file `exam.xml` afterwards (at the end of the exam, because it will be extended in later exercises) here:

Lösung

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE tdf SYSTEM "exam.dtd">
<tdf>
 <!-- town/mountain/pass could be simplified as place,
      but be careful with the name of the subelements of <stage> -->
 <town id="paris" name="Paris" country="F" elevation="30"/>
 <town id="brest" name="Brest" country="F" elevation="10"/>
 <town id="landerneau" name="Landerneau" country="F" elevation="10"/>
 <town id="nizza" name="Nice" country="F" elevation="10"/>
 <town id="sorgues" name="Sorgues" country="F" elevation="20"/>
 <town id="malaucene" name="Malaucene" country="F" elevation="300"/>
 <town id="bedoin" name="Bedoin" country="F" elevation="300"/>
 <town id="ceret" name="Ceret" country="F" elevation="120"/>
 <town id="andorra" name="Andorra" country="AND" elevation="1011"/>
 <town id="chatou" name="Chatou" country="F" elevation="30"/>
 <mountain id="ventoux" name="Mont Ventoux" country="F" elevation="1909"/>
 <mountain id="envalira" name="Port d'Envalira" country="AND" elevation="2407"/>
 <mountain id="beixalis" name="Col de Beixalis" country="AND" elevation="1795"/>
 <!-- a top-level element type for teams in not needed -->
 <rider id="poga" name="Tadej Pogacar" country="SLO" birthday="1998-09-21"/>
 <rider id="rog" name="Primoz Roglic" country="SLO" birthday="1989-10-29"/>
 <rider id="aert" name="Wout van Aert" country="B" birthday="1994-09-15"/>
 <rider id="kuss" name="Sepp Kuss" country="USA" birthday="1994-09-13"/>
 <rider id="ving" name="Jonas Vingegaard" country="DK" birthday="1996-12-10"/>
 <rider id="elis" name="Kenny Elissonde" country="F" birthday="1991-07-22"/>
 <rider id="moll" name="Bauke Mollema" country="NL" birthday="1986-11-26"/>
 <rider id="nn" name="FAKE" country="F" birthday="1900-01-01"/>
 <tour year="2020">
   <team name="UAE" riders="poga"/>
   <team name="Jumbo-Visma" riders="rog aert"/>
   <stage nr="1" date="2020-08-29" length="156" type="hilly">
     <from id="nizza"/>
     <to id="nizza"/>
     <arrived rider="nn" duration="PT00:04:39:05"/>
   </stage>
 </tour>
 <tour year="2021">
   <team name="UAE" riders="poga ving"/>
   <team name="Jumbo" riders="rog aert ving"/>
   <team name="Trek" riders="elis moll"/>
   <stage nr="1" date="2021-06-26" length="198" type="hilly">
     <from id="brest"/>
     <to id="landerneau"/>
     <arrived rider="nn" duration="PT00:03:46:23"/>
   </stage>
```

```
 <rider id="ving" name="Jonas Vingegaard" country="DK" birthday="1996-12-10"/>
 <rider id="elis" name="Kenny Elissonde" country="F" birthday="1991-07-22"/>
 <rider id="moll" name="Bauke Mollema" country="NL" birthday="1986-11-26"/>
 <rider id="nn" name="FAKE" country="F" birthday="1900-01-01"/>
 <tour year="2020">
   <team name="UAE" riders="poga"/>
   <team name="Jumbo-Visma" riders="rog aert"/>
   <stage nr="1" date="2020-08-29" length="156" type="hilly">
     <from id="nizza"/>
     <to id="nizza"/>
     <arrived rider="nn" duration="PT00:04:39:05"/>
   </stage>
 </tour>
 <tour year="2021">
   <team name="UAE" riders="poga ving"/>
   <team name="Jumbo" riders="rog aert ving"/>
   <team name="Trek" riders="elis moll"/>
   <stage nr="1" date="2021-06-26" length="198" type="hilly">
     <from id="brest"/>
     <to id="landerneau"/>
     <arrived rider="nn" duration="PT00:03:46:23"/>
   </stage>
   <stage nr="11" date="2021-07-07" length="199" type="mountains">
     <from id="sorgues"/>
     <place id="ventoux"/>
     <place id="malaucene"/>
     <place id="bedoin"/>
     <place id="ventoux"/>
     <to id="malaucene"/>
     <arrived rider="aert" duration="PT05H17M43S"/>
     <arrived rider="elis" duration="PT05H18M57S"/>
     <arrived rider="moll" duration="PT05H18M57S"/>
     <arrived rider="poga" duration="PT05H19M21S"/>
     <arrived rider="ving" duration="PT05H19M21S"/>
   </stage>
   <stage nr="15" date="2021-07-11" length="191" type="mountains">
     <from id="ceret"/>
     <place id="envalira"/>
     <place id="beixalis"/>
     <to id="andorra"/>
     <arrived rider="kuss" duration="PT05H12M06S"/>
     <arrived rider="elis" duration="PT05H16M17S"/>
   </stage>
   <stage nr="21" date="2021-07-18" length="108" type="flat">
     <from id="chatou"/>
     <to id="paris"/>
     <arrived rider="aert" duration="PT02H39M37S"/>
     <arrived rider="poga" duration="PT02H39M37S"/>
     <arrived rider="ving" duration="PT02H39M37S"/>
     <!-- intentionally, the others are not listed as an example for query 4 -->
   </stage>
 </tour>
</tdf>
```

Exercise 2 (DTD [15 Points])
Develop the DTD for your document developed in Exercise 1, use the file `exam.dtd`.
Use one of the calls

```
xmllint -loaddtd -valid --noblanks -noout exam.xml
saxonValid.bat -s:exam.xml
```

for validating it (note that xmllint provides better error messages).
Copy-and-paste the DTD from the file `exam.dtd` afterwards here:

Lösung

```
<!ELEMENT tdf (town*, mountain*, rider*, tour*)>
<!ELEMENT town EMPTY>
<!ELEMENT mountain EMPTY>
<!ATTLIST town id ID #REQUIRED
               name CDATA #REQUIRED
               elevation CDATA #REQUIRED
               country CDATA #REQUIRED>
<!ATTLIST mountain id ID #REQUIRED
               name CDATA #REQUIRED
               country CDATA #REQUIRED
               elevation CDATA #REQUIRED>
<!ELEMENT rider EMPTY>
<!ATTLIST rider id ID #REQUIRED
               name CDATA #REQUIRED
               country CDATA #REQUIRED
               birthday CDATA #REQUIRED>
<!ELEMENT tour (team*, stage+)>
<!ATTLIST tour year CDATA #REQUIRED>
<!ELEMENT team EMPTY>
<!ATTLIST team name CDATA #REQUIRED
               riders IDREFS #REQUIRED>
<!ELEMENT stage (from, place*, to, arrived*)>
<!ATTLIST stage nr CDATA #REQUIRED
                date CDATA #REQUIRED
                length CDATA #REQUIRED
                type (flat|hilly|mountains) #REQUIRED>
<!ELEMENT from EMPTY>
<!ATTLIST from id IDREF #REQUIRED>
<!ELEMENT place EMPTY>
<!ATTLIST place id IDREF #REQUIRED>
<!ELEMENT to EMPTY>
<!ATTLIST to id IDREF #REQUIRED>
<!ELEMENT arrived EMPTY>
<!ATTLIST arrived rider IDREF #REQUIRED
                  duration CDATA #REQUIRED>
```

Exercise 3 (XPath (a) [2 Points])
Use your `exam.xml` XML file as a basis for solving this and the following exercises.
None of the results should contain duplicates.

Give an XPath query or an XQuery query that returns the *names* of those riders who participated in the 2021 tour.
Write the query string in the file `query1.xq` and call it with

```
saxonXQ.bat -s:exam.xml query1.xq
```

Copy-and-paste the query from query1.xq afterwards here:

Lösung

```
//tour[@year='2021']/team/id(@riders)/@name/string()
```

Exercise 4 (XPath (b) [4 Points])
Give an XPath query or an XQuery query that returns the *years* of those tours that started/passed or finished on or over the *Mount Ventoux*.
Write the XPath query string in the file `query2.xq` and call it with

```
saxonXQ.bat -s:exam.xml query2.xq
```

Copy-and-paste the query from query2.xq afterwards here:

Lösung
Note: since "to" is used as a keyword in XPath, e.g., `//country[position() = 5 to 45]`, most XPath processors warn and ask explicitly to use `./to`.

```
//tour[stage/(from|place|./to)/id(@id)/@name="Mont Ventoux"]/@year/string()
```

Exercise 5 (XPath/XQuery (c) [6 Points])
Give an XPath query or an XQuery query which, for every rider, yields the total number of stage victories he ever had in the Tour de France. The results should be ordered by the number of victories descending in the form

```
<rider name="..." number="..."/>
```

Copy-and-paste the query from query3.xq afterwards here:

Lösung

```
(:returns for all riders incl non-winners :)
for $r in //rider
let $num := count(//stage[arrived[1]/@rider = $r/@id])
order by $num descending
return
<rider name="{string($r/@name)}" number ="{$num}"/>
```

```
(: returns only at-least-once-winners :)
for $win in //stage/arrived[1]
group by $rider := $win/@rider
return
<rider name ="{id($rider)/@name}" number="{count($win)}"/>
```

Exercise 6 (XPath/XQuery (d) [6 Points])
Give an XPath query or an XQuery query that yields the names of all riders that *at least once finished* a stage *passing* the Mont Ventoux, but never finished the *last* stage of a

tour (this stage usually leads to Paris, but also other intermediate stages might have their destination in Paris).
Copy-and-paste the query from query4.xq afterwards here:

Lösung

```
//stage[(place|arrival)/id(@id)/@name="Mont Ventoux"]/arrived/id(@rider)
  (: the riders nodes, so no distinct-values needed :)
      [not (@id = //tour/stage[last()]/arrived/@rider)]
  /@name/string()
```

or very explicit by XQuery:

```
for $r in //rider
where $r/@id =
    //stage[(place|arrival)/id(@id)/@name="Mont Ventoux"]/arrived/@rider
  and not ($r/@id = //tour/stage[last()]/arrived/@rider)
return $r/@name/string()
```

Note: comparing "." (i.e. a rider's element node) by "=" would compare the empty strings.

Exercise 7 (XPath/XQuery (e) [6 Points])
Give an XQuery query or an XPath query that returns those countries/country codes from which in *every year since 1990* at least one rider participated in the tour.
Copy-and-paste the query from query5.xq afterwards here:

Lösung

```
for $c in distinct-values(//rider/@country)
where every $y in //tour/@year[. >=1990]
satisfies //tour[@year = $y]/team/id(@riders)/@country = $c
return $c
```

Exercise 8 (XPath/XQuery (f) [9 Points])
Give an XQuery query that returns for the 2021 tour the total result, i.e., for every rider who finally completed the last stage, the total time he needed. The results should be of the format

```
<rider name="..." totalduration="..."/>,
```

ordered by the total time ascending. The query must allow that for evaluating the result for other years, only the year has to be changed.
Copy-and-paste the query from query6.xq afterwards here:

Lösung

```
for $r in //tour[@year="2021"]/stage[last()]/arrived/id(@rider)
let $total :=
  sum(//tour[@year="2021"]/stage/arrived[@rider = $r/@id]/
      xs:dayTimeDuration(@duration))
order by $total ascending
return <rider name="{$r/@name}" totalduration="{$total}"/>
```

Exercise 9 (XPath/XQuery (g) [9 Points])
Give an XQuery query or an XSLT stylesheet that, for each rider R, returns a list (without duplicates) of all mountains or places P higher than 1000m that he reached during his Tour de France participations. For each such rider R, the result should be of the form

```
<result rider="name-of-R">
   name-of-place-P1 ... name-of-place-Pn
</result>
```

(no duplicate P's, in any order, with arbitrary whitespaces)

Copy-and-paste the query from query7.xq afterwards here:

Lösung

```
for $r in //rider
let $places :=
    //stage[arrived/@rider = $r/@id]
       /(from|./to|place)/id(@id)[@elevation > 1000]/@name
return <result>
  {  $r/@name,
     distinct-values($places)
  }
  </result>
```

Exercise 10 (XSLT [15 Points])
Extend the given XSL stylesheet frame `exam.xsl` to an XSLT stylesheet that returns for the 2021 tour a simple HTML page that contains for every rider a table which lists his name, and then his positions in each of the stages that he finished.

Use the following call to execute it:

```
saxonXSL.bat -s:exam.xml exam.xsl
```

Copy-and-paste the XSLT stylesheet from exam.xsl afterwards here:

Lösung

(1) Solution by using position() wrt. the current context:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="2.0">

 <xsl:template match="tdf">
   <html> <body>
      <xsl:apply-templates select="tour[@year=2021]/team/id(@riders)"/>
   </body> </html>
 </xsl:template>

 <xsl:template match="rider">
    <table>
     <tr><td colspan="2">
      <xsl:value-of select="@name"/>
     </td></tr>
     <xsl:apply-templates select="//tour[@year=2021]/stage">
       <xsl:with-param name="riderid" select="@id"/>
     </xsl:apply-templates>
   </table>
 </xsl:template>

 <xsl:template match="stage">
   <xsl:param name="riderid"/>
   <xsl:for-each select="arrived">
     <!-- now, the context are all <arrived> elements of this stage: -->
     <xsl:if test="@rider = $riderid">
       <tr>
       <td>Stage: <xsl:value-of select="../@nr"/></td>
       <td>Pos: <xsl:value-of select="position()"/></td>
          <!-- the position wrt.\ the context yields the rank-->
       </tr>
     </xsl:if>
   </xsl:for-each>
 </xsl:template>
</xsl:stylesheet>
```

(2) Solution using index-of:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="2.0">

 <xsl:template match="tdf">
   <html> <body>
      <xsl:apply-templates select="tour[@year=2021]/team/id(@riders)"/>
   </body> </html>
 </xsl:template>

 <xsl:template match="rider">
    <table>
     <tr><td colspan="2">
      <xsl:value-of select="@name"/>
     </td></tr>
     <xsl:apply-templates
          select="//tour[@year=2021]/stage/arrived[@rider = current()/@id]"/>
   </table>
 </xsl:template>

 <xsl:template match="arrived">
    <tr>
     <td>Stage: <xsl:value-of select="../@nr"/></td>
     <td>Pos: <xsl:value-of select="index-of(../arrived/@rider,@rider)"/></td>
    </tr>
   </xsl:template>
</xsl:stylesheet>
```

As shown in (1) running the third template over the stages, and then to look for the rider's result requires to communicate the rider's name as an xsl:param. Then, an xsl:if is needed, and then, position() can be used.

Exercise 11 (Miscellaneous (a) [4 Points])
Consider that the same task/database should be realized by a relational database using SQL.
What would be a central difficulty in contrast to XML?
Which query/queries from above would be much more complicated?

Lösung

- The data items in a relational database are not ordered. So, the sequences of (i) stages, (ii) places on a stage, and (iii) riders arriving would require explicit tuples (number, value).
- Mainly, queries (c) (winners of stages, but this index would always be =1) and even more (d) and (f) (last stage to Paris, can be 19th, 20th, 21st) and the XSLT (finding out the position of a rider) would require rewriting.

Exercise 12 (Miscellaneous (b) [4 Points])
Consider that the database should be extended as follows: for every town/mountain, an HTML page/tree with some touristic information should be stored.
Describe (shortly) how to extend the XML and the DTD.

Lösung

- The HTML fragment is stored as the content of the -up to no empty- town/mountain elements (note that this is just a nested tree of HTML/XHTML elements):

```
<town id="paris" name="Paris" country="F" elevation="30">
  <html><head>...</head><body>...</body></html>
</town>
```

- for the DTD, set

```
<!ELEMENT town (html?)>
<!ELEMENT mountain (html?)>
```

  and import the HTML DTD.

The below example shows how it works (additionally using namespaces):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE tdf SYSTEM "exam2.dtd">
<tdf xmlns="http://www.semwebtech.org/tdf/">
 <!-- town/mountain/pass could be simplified as place,
      but be careful with the name of the subelements of <stage> -->
 <town id="paris" name="Paris" country="F" elevation="30">
   <html> <!-- the xmlns is FIXED in the html DTD, so not needed here:
                xmlns="http://www.w3.org/1999/xhtml" -->
     <head><title>Paris</title></head>
     <body><h1>The French Capital: Paris</h1></body>
   </html>
 </town>
 <town id="nizza" name="Nice" country="F" elevation="10">
   <html>
     <head><title>Nizza/Nice</title></head>
     <body><h1>Cote d'Azur: Nice</h1></body>
   </html>
 </town>
 <rider id="poga" name="Tadej Pogacar" country="SLO" birthday="1998-09-21"/>
 <rider id="rog" name="Primoz Roglic" country="SLO" birthday="1989-10-29"/>
 <rider id="aert" name="Wout van Aert" country="B" birthday="1994-09-15"/>
 <tour year="2020">
   <team name="UAE" riders="poga"/>
   <team name="Jumbo-Visma" riders="rog aert"/>
   <stage nr="1" date="2020-08-29" length="156" type="hilly">
     <from id="nizza"/>
     <to id="nizza"/>
     <arrived rider="poga" duration="PT00:04:39:05"/>
   </stage>
 </tour>
</tdf>
```

The DTD contains a reference to the HTML DTD (cheat slightly, the original W3C DTD cannot be referenced due to the issues described on the lecture slides; cf. catalogue usage):

```
<!ELEMENT tdf (town*, mountain*, rider*, tour*)>
  <!ATTLIST tdf xmlns CDATA #IMPLIED>
<!ELEMENT town (html?)>

<!ENTITY % htmldtd SYSTEM 'myxhtml.dtd'>
%htmldtd;

<!ELEMENT mountain (html?)>
<!ATTLIST town id ID #REQUIRED
               name CDATA #REQUIRED
               elevation CDATA #REQUIRED
               country CDATA #REQUIRED>
<!ATTLIST mountain id ID #REQUIRED
               name CDATA #REQUIRED
               country CDATA #REQUIRED
               elevation CDATA #REQUIRED>
<!ELEMENT rider EMPTY>
<!ATTLIST rider id ID #REQUIRED
               name CDATA #REQUIRED
               country CDATA #REQUIRED
               birthday CDATA #REQUIRED>
<!ELEMENT tour (team*, stage+)>
<!ATTLIST tour year CDATA #REQUIRED>
<!ELEMENT team EMPTY>
<!ATTLIST team name CDATA #REQUIRED
               riders IDREFS #REQUIRED>
<!ELEMENT stage (from, place*, to, arrived*)>
<!ATTLIST stage nr CDATA #REQUIRED
                date CDATA #REQUIRED
                length CDATA #REQUIRED
                type (flat|hilly|mountains) #REQUIRED>
<!ELEMENT from EMPTY>
<!ATTLIST from id IDREF #REQUIRED>
<!ELEMENT place EMPTY>
<!ATTLIST place id IDREF #REQUIRED>
<!ELEMENT to EMPTY>
<!ATTLIST to id IDREF #REQUIRED>
<!ELEMENT arrived EMPTY>
<!ATTLIST arrived rider IDREF #REQUIRED
                  duration CDATA #REQUIRED>
```

For the file "myxhtml.dtd", download `https://www.w3.org/TR/xhtml1/dtds.html#a_dtd_XHTML-1.0-Strict` and remove the contained entity references to auxiliary files at the beginning.

The sample query illustrates the use of the namespaces:

```
(: saxonXQ -s:exam2.xml exam2.xq :)
declare namespace t="http://www.semwebtech.org/tdf/";
declare namespace ht="http://www.w3.org/1999/xhtml";
/t:tdf//t:town//ht:title
```

The following frames can be used:

- Frame for XML file `exam.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE put-name-here SYSTEM "exam.dtd">
  to be extended here
```

- Frame for XML stylesheet `exam.xsl`:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="2.0">
  to be extended here
</xsl:stylesheet>
```