School of Engineering and Computer Science

SWEN 432
Advanced Database Design and Implementation

Assignment 1

Due date: Friday 28 March at 11:59 pm

The objective of this assignment is to test your understanding of XML documents, DTD and XML Schema schemas.

The Assignment is worth 4.0% of your final grade. The Assignment is marked out of 100.

When doing the assignment you will need to use the xmllint XML parser to check whether your XML documents are:

- Well formed, or
- Valid with regard to a DTD, or XML Schema schema.

You can use xmllint from UNIX shell. To do so, type

```
>xmllint
```

To get information how to use xmllint, type

```
>man xmllint
```

Include your commands and xmllint messages from the shell prompt in your answers.

You may also use other XML parsers. Check carefully whether the other parser returns the same diagnostics as xmllint.

Question 1. `WRONG_Boat_Hire.xml` [10 marks]

Pavle made the file `WRONG_Boat_Hire_14.xml`. The document is given at the Assignments page of the course web page as a `.txt` file, since a web browser automatically parses an XML file and, if not well formed, refuses to display it. When `xmllint` parsed the document, it returned so many error messages that Pavle decided to ask you to help him in fixing the errors and producing a well formed XML document.

a) **[5 marks]** In your answer, alter the `WRONG_Boat_Hire_14.txt` document by commenting each of its errors. Your comments should point to the real cause of each error, since the diagnostices made by `xmllint` may be missleading. Also, produce the needed `Boat_Hire_WellFormed_14.xml` document. Submit both:

   i. The commented `WRONG_Boat_Hire_14.txt` and
   ii. `Boat_Hire_WellFormed_14.xml` documents.

b) **[5 marks]** The Assignments web page also contains the `Boat_Hire_14.dtd` schema. Use it to check whether your `Boat_Hire_WellFormed_14.xml` is valid with regard to the `Boat_Hire_14.dtd` schema. If it is not, correct and comment the errors in a new XML document under the name `Boat_Hire_Valid_14.xml`.

Question 2. [20 marks]

Assume a boat can be driven only by:

- Rows,
- Sail,
- Motor,
- Rows and Sail,
- Rows and Motor, and
- Sail and Motor.

Your task is to make a new DTD `Boat_Hire_Extended_14.dtd` starting from the `Boat_Hire_14.dtd` given in question 1. To help you accomplish your task, a fragment of the new DTD is given below. The fragment tells you how to extend the content model of the `Boat` element and to use empty elements to represent the three basic boat driving forces. Defining the content model of the `Driven_By` element is left to you. Your new DTD has to validate only those files where boats are driven by one of the ways listed above.

```
<!ELEMENT Boat (Number, Color, Driven_By, Reserves*)>
<!ELEMENT Number (#PCDATA)>
<!ELEMENT Color (#PCDATA)>
<!ELEMENT Driven_By (you to define)>
<!ELEMENT Row EMPTY>
<!ELEMENT Sail EMPTY>
<!ELEMENT Motor EMPTY>
<!ATTLIST Boat name CDATA #REQUIRED>
<!ELEMENT Reserves (Date)>
<!ELEMENT Date (#PCDATA)>
<!ATTLIST Reserves ref_sail IDREF #REQUIRED>
```

Question 3. SQL Queries as XML Documents [35 marks]

Consider the following simplified description of a SQL query:

- A SQL query starts by a `SELECT` close,
- A `SELECT` close contains either
  - An arithmetic expression (`arith_exp`), or
  - A `list_of_col_names`, followed by:
    - A compulsory `FROM` close, and
    - An optional `WHERE` close, and
    - An optional GROUP BY (`GROUP_BY`), and
    - An optional `HAVING` close, and
    - An optional ORDER BY (`ORDER_BY`) close,
  - An arithmetic expression (`arith_exp`) contains text,
  - A `list_of_col_names`, contains at least one `COL_NAME`,
    - A `COL_NAME` contains text,
  - A `FROM` close contains a non empty list whose elements are:
    - Either a table name (`TAB_NAME`), or
    - A (`SELECT, ALIAS`) pair

    in no particular order
    - An `ALIAS` contains text that is the short name of a temporary table produced by SELECT,
  - A `WHERE` close contains:
    - At least one `condition`, but if there are more than one `condition`, then each `condition` except the first one, is preceded by a `log_operator`,
    - A `condition` contains:
      - A `COL_NAME`,
      - An `OPERATOR`, and
      - Either a `COL_NAME`, or a `VALUE`, or a `SELECT` close,
    - A `log_operator` contains text with two possible values `AND` and `OR`,
    - An `OPERATOR` contains text with a value from the set {=, <, >,…, `IN`, `ANY`,… },
  - A `GROUP_BY` close contains a `list_of_col_names`,
  - A `HAVING` close contains a `condition`,
  - An `ORDER_BY` contains a `list_of_col_names` and has `sort_order` that is text with two possible values `ASC` and `DESC`, where `ASC` is default.

a) **[20 marks]** Your first task is to use the description above and make a DTD for storing SQL queries as XML documents. Use words in `Courier New` font in the text above to name concepts (elements, attributes, …) in your DTD. Call your DTD `SQL_Query_14.dtd`.

b) **[2 marks]** Consider the SQL query below. Make an XML document that will contain the query and be valid with regard to your `SQL_Query_14.dtd`.

```
SELECT 2+2;
```

c) **[5 marks]** Consider the SQL query below. Make an XML document that will contain the query and be valid with regard to your `SQL_Query_14.dtd`.

```
SELECT s.StudentId, First_Name, Surname

FROM Student s

WHERE s.StudentId IN

(SELECT StudentId FROM Class WHERE code = “SWEN432”);
```

d) **[8 marks]** Consider the SQL query below. Make an XML document that will contain the query and be valid with regard to your `SQL_Query_14.dtd`.

```
SELECT s.StudentId, First_Name, Surname, Title, Grade

FROM Student s, (SELECT StudentId, Title, Grade FROM Course
c Class g WHERE c.Code = “SWEN432” AND Year = 2014 AND
c.Code = g.Code) cg

WHERE s.StudentId = cg.StudentId

ORDER BY s.StudentId;
```

Note:

When making XML documents containing SQL queries, you may need to prefix some column names (as the content of `COL_NAME`) by their table’s short name (alias) using dot notation.

The value of `TAB_NAME` is a text consisting of a table’s name followed by a space and table’s short name (alias).

Question 4. Mapping a DTD to XML Schema [10 marks]

Map `Boat_Hire_14.dtd` given in question 1 into an XML Schema. Your mapping should preserve all structural information and constraints contained in `Boat_Hire_14.dtd`. Your `Boat_Hire_Valid_14.xml` should validate with the XML Schema you produced by mapping.

Question 5. Structural Constraints [25 marks]

In this question, you are asked to deal with structural constraints.

a) **[3 marks]** LECTURER is a child of the CLASS element. Use DTD syntax to define the following structural constraint: “a class has at most two lecturers, but there may be a class having no lecturers nominated yet”.

b) **[18 marks]** Let Q, A, and P be element names. What is the difference between the following two DTD expression:

i. [6 marks] <! ELEMENT Q (A, P)?> and <! ELEMENT Q (A?, P?)>?

ii. [6 marks] <! ELEMENT Q (A, P)+> and <! ELEMENT Q (A+, P+)>?

iii. [6 marks] <! ELEMENT Q (A | P)+> and <! ELEMENT Q (A*, P*)>?

In your answer, translate each of regular expressions into English paying attention to rigor of your translation. Compare them and make a conclusion. Use XML fragments to support your claims. Note, use of XML fragments as examples that justify your claims will influence marks allocated to your answers.

c) **[4 marks]** Let F, C, L, A, and S be element names. Use the syntax of XML Schema to translate the following DTD fragment:

<! ELEMENT F (C+, L, A?, S*, (L | S)+)+ >

into an XML Schema fragment.

Submission Instruction:

Submit your answers to assignment questions via the school electronic submission system and hand-in a printed version in the hand-in box on the second floor of the Cotton Building.

Submit all your `.xml`, `.xsd` and `.dtd` files electronically (including the files you used for testing), so that they can be tested using `xmllint`.