XML and Databases
COMP 4317/9317
Final Exam (open book) --- 23rd June 2009

(1)**[4]** Consider this DTD:

```
<!DOCTYPE bib [
  <!ELEMENT bib (book | journal)*>
  <!ELEMENT book (author, title)>
  <!ELEMENT journal (author, title, cites?)>
  <!ELEMENT cites (book | journal)*>
  <!ELEMENT author (#PCDATA)>
  <!ELEMENT title (#PCDATA)>
  <!ATTLIST book isbn ID #REQUIRED>
]>
```

This DTD is included in each of the following. Say for each whether or not it is well-formed XML (with respect to the DTD!). If it is not well-formed, explain ***all*** violations that you can find.

```
a) <bib><book></book></bib>
b) <bib><journal isbn=”xyz”><author/><title/></journal></bib>
c) <bib><book
isbn=”123”><author/><title/></book><journal><author/><title/><cites><
book isbn=”123”><author/><title/><book/></cites></journal></bib>
d) <bib book=”isbn”></bib>
e) <bib>no entries</bib>
f) <bib><journal><author/><title/><!—- all empty>>->--></bib>
g) <bib></bib><bib></bib>
h) <bib><author></author><title></title></Bib>
```

(2)**[4]** Consider again the DTD from number (1).
If a journal or book subtree appears below a cites-node, then we say that this journal or book is being cited.
(a) Write pseudo code that uses DOM and prints each journal and book that is being cited, together with the number of times it is cited.
(b) Is it possible, with the DTD of (1), that a book cites itself? Explain. Do you see a better way of citing, using attributes? How?

(3)**[4.5]** Given a DAG as
dag(node id)=List(node id’s) and label(node id)=String
(a) write pseudo code that prints in XML format the tree that is represented by the dag. For instance, if 1:a, 2:b[1,1,1] is your dag, then your code should print <b><a></a><a></a><a></a></b>
(b) given two dags, dag1 and dag2 (both not necessarily minimal!), write pseudo code that checks whether the trees represented by dag1 and dag2 are equal. Your program should NOT decompress both dags, and then check equality of the strings; instead, your program should run in linear time with respect to the sum of sizes of dag1 and dag2!

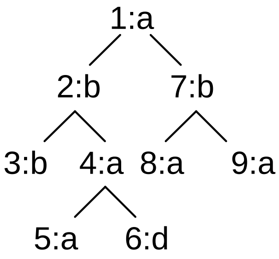
(4)**[4]** Consider a (pre,post) table: Given a pre-order number x, the mapping post(x) returns the post-order of the node whose pre-order is x. Write pseudo code that, for a node p, prints pre-numbers of
a) its descendants
b) its children
c) its following-siblings
d) all following nodes that are leaves
e) all nodes that are at least two edges away from p
f) Given a sequence of nodes p_1,…,p_n in pre-order, how can you compute in an optimal way all the preceding nodes of p_1,…,p_n.

(5)**[8]** Again, under the DTD of (1), but without the isbn-attribute. Write XPath queries that select

a) all journal nodes which do not have a cites-child
b) all title nodes that appear below journal nodes
c) the right-most leaf
d) the deepest node of the tree (right-most one, if not unique)
e) all authors which have written a journal and a book (i.e., author-nodes that appear under journal and book nodes, with the same text-string below them)
f) all books which are cited (based on their title)
g) same as f, but each book printed only once, in pre-order
h) all author names which appear inside the title of a book

(6)**[4.5]** For the tree T on the right, write numbers of nodes selected by the following XPath expressions.

a) //a
b) /*//*//a[preceding::a]
c) //*[.//d]
d) /*[not(a and b)]
e) //*[count(.//*)=count(ancestor::*)]
f) //c[position()=last()]
g) /descendant:*[position() mod 2 = count(.//*)]
h) //*[count(*)>1 and not(child::*[not(self::a)])]
i) //*[preceding-sibling::b]



7)**[2]** Consider the tree T in (6). Show in detail the *bottom-up evaluation* for the query Q = //*[not(ancestor::b)]. First, give for Q the corresponding evaluation tree over ∩, ∪ , lab(b), child, descendant, etc. Then show the actual subsets of { 1, 2, …, 9 } which are selected by the different nodes of the evaluation tree.

8)**[2]** Show the “KMP-automaton” for //a/b/*/*/a   and show the automaton run on the input abababa.

9)**[2]** Using the canonical model, show that p is contained in q, for
p = /a[.//b[c/*//d]/b[c//d]/b[c/d]]
q = /a[.//b[c/*//d]/b[c/d]]

10)**[5]** a) Why is the Glushkov automaton important for DTDs? How is it used to check whether an XML document is valid for a given DTD?
b) Give the Glushkov automaton for E = (a? b? c?)*
c) How many edges, in terms of m, does the Glushkov automaton have for the expression (a_1? a_2? a_3? … a_m?)* ?
d) For a deterministic expression of length m and an input of length n, how much time is needed to check the input against the expression? How is this different for general (non deterministic) expressions?
e) It is known that no equivalent deterministic expression exists for E=(a|b)*a(a|b). Show two expressions E1 and E1 such that (E1 | E2) is equivalent to E. (This proves that det. expressions are not closed under union).

Total: **40** Points

Good luck and best success with this exam!