XML and Databases
COMP 4317/9317
Final Exam (open book) --- 11th June 2008

(1)**[4]** For each of the following, explain whether or not it is well-formed XML. Explain all violations that you find. (Watch out, some of these might be well-formed)
a) <comment>For numbers x with x<>5, x/5 is not 1.</comment>
b) <auto<node>>XF23414</auto<node>>
c) <b><b><b at="7"/><b at="7"><b/></b><b/></b at="4"></b>
d) <inside att="blah<!--a comment--> EOF"/>
e) <a a="a"/>
f) <_a><!-->--></_a>
g) <h><!-- anything here:a-z, .. --></h>
h) <a><a/><b></b><c></c>

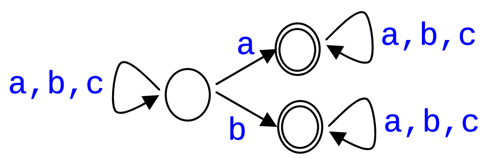
(2)**[3.5]** Write pseudo code that uses DOM access to *iteratively* print all text nodes of a document, in reverse document order (i.e., from right-to-left in terms of the document tree). You may not use recursion!

(3)**[3]** Write pseudo code that, given a DAG counts how many a-nodes it has, using only one run through the DAG table (every row is visited once).
The DAG is: dag(node id)=List(node id's) and lab(node id)=String.

(4)**[3]** Explain how hashing is used to find the minimal DAG of a tree. Imagine there are only four labels: a,b,c,f and a hash table with only three buckets; find the dag for a(b(c,c),b(f,c),b(f,c),b(f,f)). For this example, what would be an optimal hash function? Explain! (how many node comparisons are saved wrt no-hash or bad hash function?)

(5)**[2.5]** Imagine a (pre,size) table, given by a mapping size; e.g., for <a><b/><b/></a> we have size(1)=2, size(2)=0, and size(3)=0. Write pseudo code that, for a node p, prints pre-numbers of
a) its descendants
b) its children
c) its parent
d) its following-siblings
e) its preceding nodes.

(6)**[4]** Consider the following automaton A:

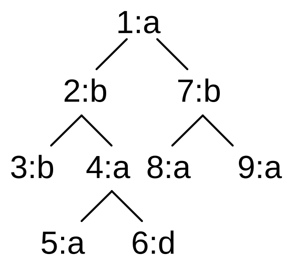


a) Show a string accepted by A, and one that is rejected.
Is A deterministic? Give an equivalent deterministic automaton B.
b) Give a regular expression for the strings accepted by A
c) Is your expression from b) 1-unambiguous? Show the Glushkov automaton.
d) Give a 1-unambiguous expression for the strings over a,b which do not contain the substring aa and do not end on a.

(7)**[8]** Write XPath queries that select
a) all element nodes which have no text children
b) all element nodes which have an a-attribute

c) all element nodes at level 100
d) all element nodes which have 2 attributes with different values
e) the node with the smallest attribute value
f) the next-sibling of each a-node
g) the left-most leaf (element) node of the document
h) all odd children of a-nodes (1st child, 3rd, 5th, etc)

(8)**[4]** For the tree on the right, write numbers of nodes selected by the following XPath expression.
a) /a//b
b) /descendant::a[3]/following::*[2]
c) //a/b
d) //a[parent::*//a]
e) //*[not(following::*)]
f) //*[count(ancestor::*)=2]
h) /*/*//*
i)//*[count(preceding::*)>count(following::*)]



(9)**[2]** Explain how the XPath expression EX=//a/b/*/b/a can be evaluated on an XML stream.
How much memory do you need?
a) if you print node numbers
b) if you print the subtrees at selected nodes.
Explain!

10)**[3]** Given four nodes in the (pre,post)-plane: (p1,o1),..,(p4,o4):
a) Write an SQL query which computes (duplicate-free and in pre-order) the following-nodes of the four nodes (p1,o1) up to (p4,o4).
b) Can you find a query that returns duplicate free answers, but does not use the DISTINCT instruction? Explain.

11)**[3]** a) Give XPath expression p and q such that p1 0-contained in p2, but not 1-contained.
Give p and q such that p1 1-contained in p2, but not 2-contained.
b) explain why 0- and 1-containment are the same for XPath expression that only use child and descendant axes.
c) Is p 0-contained in q, for
p=/r//a[parent::*/b] and q=/r//a[following:b]?

```
[1]   document   ::= prolog element Misc*
[2]       Char   ::= a Unicode character
[3]          S   ::= (' ' | '\t' | '\n' | '\r')+
[4]   NameChar   ::= (Letter | Digit | '.' | '-' | ':'
[5]       Name   ::= (Letter | '_' | ':') (NameChar)*
[14]  CharData   ::= [^<&]* - ([^<&]* ']]>' [^<&]*)
[15]   Comment   ::= '<!--' ((Char - '-') | ('-' (Char - '-')))* '-->'
[25]        Eq   ::= S? '=' S?
[39]   element   ::= EmptyElemTag
                     | STag content Etag
[40]      STag   ::= '<' Name (S Attribute)* S? '>'
[41] Attribute   ::= Name Eq AttValue
[10]   AttValue  ::= '"' ([^<&"]|Reference)* '"' | "'" ([^<&']|Reference)* "'"
[42]      ETag   ::= '</' Name S? '>'
[43]   content   ::= CharData? ((element|Reference|CDSect|PI|Comment) CharData?)*
[44]EmptyElemTag ::= '<' Name (S Attribute)* S? '/>'
[84] Letter      ::= [a-zA-Z]
[88] Digit       ::= [0-9]
```

Good luck and best success with this exam!