1. (6 points) Of the following statements, identify all that hold about names, identifiers, and namespaces.
   A. The main prerequisite for a unique identifier scheme is the existence of an agreement in the community whose members would use that identifier scheme
   B. Internet domains (such as ncsu.edu) are not valid identifiers because an architecture exists to resolve them
   C. We can use URIs to identify namespaces, which themselves consist of identifiers

Solution:

A: as with the example of MAC addresses—if vendors begin to use each other's codes, they would compromise the MAC naming scheme

B is false: the existence or implementation of an architecture doesn't hurt an identifier scheme

C

2. (22 points) Of the following statements, identify all that hold about metadata, XML, and XML Schema.
   A. XML InfoSet specifies that there is exactly one document root in an XML document
   B. A well-formed XML document can be a valid (executable) XQuery query
   C. The output produced by an XQuery query must be a well-formed XML document
   D. Metadata is useful because it enables data to be exchanged without prior agreement about the specific data being exchanged
   E. XML is useful because it removes the need for metadata
   F. XML supports data sharing "across time," meaning that XML documents that are read a long time after they are prepared are easier to comprehend than documents stored in proprietary formats
   G. XML is well-suited for representing purchase orders, but not for representing product descriptions and catalogs
   H. A major limitation of XML that it structures documents as trees and thus fails to represent graphs in general
   I. In XML, <foo/> is an example of a nil element
   J. In XML, whitespace characters are not included in a text node
   K. In any XML document that is valid with respect to an XML Schema, we cannot have an element occurring as a subelement of another copy of the same element; that is, <foo><foo></foo></foo> is not valid for any schema

Solution:

A

B

C is false

D is true because it removes the need for prior agreement about whatever it describes. This doesn't mean that the metadata describes everything that could be described. And it doesn't mean that the metadata itself should not or cannot have additional metadata describing it

E is false: we need metadata even when we use XML

F

G is false: XML can represent anything that can be computationally represented

H is false: XML can represent anything that can be computationally represented

I is false: nil is expressed via an attribute xsi:nil given a value "true"

J is false

K is false

3. (16 points) Of the following statements, identify all that hold about XPath. (Below, E is an arbitrary XPath expression; i and j are positive integers.)
   A. The notation // abbreviates /descendant-or-self::node()/
   B. For an arbitrary XPath expression E, E[-1] is empty
   C. There are no variables in XPath expressions
   D. The XPath construct for accessing a parent is worthless since to get to a specific node, you would have to pass its parent anyway
   E. Not every node has a child
   F. Every node has a parent
   G. If Node A is among the preceding-siblings of Node B, then B is among the following-siblings of Node A
   H. XPath constructs for navigating documents include symbolic links analogous to those in leading file systems

Solution:

A

B is true: it is tempting to think in terms of the Effective Boolean Value of -1. However, EBV applies only when you have a nonnumeric or something like a Boolean formula. When i is an integer between 1 and last(), the $[i]$ construct selects the specified member. When i is some other number, the $[i]$ construct selects nothing. When i is something else, the $[i]$ construct selects the members for which i computes to true.

C

D is false: you might need to look at the parent for purposes of selection; also you might get to a node via some axis (such as descendant-or-self or following) in which the parent is not explicitly mentioned

E: leaves don't have children

F is false: the document root has no parents

G

H is false: there are no symbolic links in XPath as such

4. (24 points) Of the following statements, identify all that hold about XQuery. (Below, Set and Pred are functions and $x and $v are variables.)
   A. The snippet let $x := $x is invalid because it uses $x to define itself
   B. If some $x in Set($v) satisfies Pred($x,$v) then every $x in Set($v) satisfies Pred($x,$v)
   C. If some $x in Set($v) satisfies Pred($x,$v) returns nothing then every $x in Set($v) satisfies Pred($x,$v) may still return something
   D. In any XQuery query, there must be at least one return clause

E. In any XQuery query, there must be at most one return clause

F. The main reason to use let is that it is more efficient than for, which iterates over each element of the given set

G. Recursion is natural in XQuery queries because XML documents often exhibit a (fairly) regular structure

H. An XQuery query can read in at most one XML file (it does so using the doc construct)

I. It is possible to write nontrivial XQuery queries (such as to reverse a list) without using the return construct

J. In for $x in Sequence . . . , the expression Sequence may not refer to $x

K. It is not possible to write nonterminating queries (i.e., those with infinite loops) in XQuery

L. In for $x in Sequence . . . , the expression Sequence is not recalculated for each iteration

Solution:

A is false: the snippet is not invalid and it doesn't use $x to define itself; depends on where it is placed

B is false: when Set($v) has at least one member that satisfies Pred($x,$v) and at least one member that does not

C: when Set($v) is empty

D is false: for example, the `reverse-1.xq` listing has no returns

E is false: for example, the `test-nested-2.xq` listing has two returns

F is false: they are simply different constructs

G

H is false

I: for example, the `reverse-1.xq` listing has no returns

J is false: similar to A

K is false

L is true: this is one of the differences between the XQuery for construct and the for loops found in (conventional) imperative languages