1. (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 text() function extracts the first text node under the current element
   B. If i≠j, we never have E[i][j]=E[j][i]
   C. The only cases where E[i][j] = E[i] are when i=j=1
   D. Assuming E does *not* evaluate to (), E/foo is never equal to E[foo]
   E. XPath doesn't include recursive queries
   F. The query let $x := 1 return (2)[1] produces a result of 2
   G. In XPath, * abbreviates node()
   H. E[@lg] selects members of E for which attribute @lg is defined and is not equal to the empty string

Solution:

E is true: recursion doesn't quite make sense for XPath (recursion makes sense for XQuery, of course)

F is true: 2 is the first value of the sequence (2)

A is false: text() returns all text nodes

B is false: the two expressions are equal when i and j are larger than last()

C is false: both sides yield () when i is larger than last() in its context

D is false: E/foo selects all foo subelements for members of E whereas E[foo] selects members of E that have a subelement—and these are equal when both are empty, which is possible even when E is not empty

G is false: * stands for all elements

H is false: it just tests for the existence of an attribute even if its value is the empty string

2. (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. Using no axes other than parent and child, we can write an XQuery function that would compute the *ancestors* of its argument element
   B. The order of evaluation of bindings in XQuery's some and every clauses is implementation-dependent
   C. XQuery will become a candidate recommendation of the W3C in 2008
   D. If every $x in Set($v) satisfies Pred($x,$v) then some $x in Set($v) satisfies Pred($x,$v)
   E. The Effective Boolean Value of a proper negative fraction such as -0.5 is true
   F. The Effective Boolean Value of a string containing a proper negative fraction such as "-0.5" is neither true nor false
   G. An easy way to swap values of $x and $y is let $x := $y followed immediately by let $y := $x
   H. Consider a let clause with multiple variables. In such a clause, a positional variable (as in at $pos) refers to the position of each variable being assigned
   I. The snippet 5 is a valid XQuery query even though it is not an XML document
   J. The snippet <foo>5</bar> is a valid XQuery query even though it is not an XML document
   K. If you ever see $x in an XQuery query, and the $x is not placed within quotes, then the $x is a variable
   L. An executable XQuery query cannot contain any free variable

Solution: A, B, I, E, L

E is true: negative numerics have an effective boolean value of true

C is false: XQuery became a recommendation in January 2007

D is false: consider when Set($v) is empty

F is false: nonempty strings have an effective boolean value of true

G is false: it's not a swap

H is false: no positional variables for let

J is false: it's ill-formed

K is false: $x placed in a return is interpreted as a string even though it is not in quotes