1. (4 points) Of the following statements, identify all that hold about architecture.

   A. DoDAF specifies a number of views to capture different aspects of a system being modeled

   > **Solution:** A is true:

   B. We can abstract out the interactions and policy points from DoDAF operational views

   > **Solution:** B is true:

2. (32 points) Of the following statements, identify all that hold about metadata and XML.

   A. Electronic documents were exchanged by businesses long before XML and even before the Internet came into being

   > **Solution:** A is true:

   B. A major benefit of metadata is to facilitate the exchange of data over time

   > **Solution:** B is true:

   C. A major disadvantage of Comma Separated Values with respect to XML is that XML better enables us to express nested structure

   > **Solution:** C is true: there is no explicit nesting in CSV, but there is in XML

   D. Sometimes people can create different URIs for the same XML namespace and as a result what they think are different namespaces are treated by the XML processor as the same namespace

   > **Solution:** D is false: the URI is simply a name of a namespace; an XML processor is not expected to determine whether two URIs refer to the same namespace

   E. Using a URI that includes the domain name of another company is not only bad form, it is risky because the other company could change the resource pointed to by that URI; for this reason, a URI such as http://www.ncsu.edu/fb77 is unsuitable as an identifier for a namespace for anyone not affiliated with NCSU

   > **Solution:** E is false: which URI is used to identify a namespace is simply a matter of convention; no physical resource need exist; it means nothing other than what we want it to mean

   F. Using a URI such as http://localhost/fb77 that includes a relative address is risky because it means different things on different computers

   > **Solution:** F is false: which URI is used to identify a namespace is simply a matter of convention; no physical resource need exist; it means nothing other than what we want it to mean and has no dependence on which computer we might be thinking of

   G. Whereas in most beginner tutorials an XML document has a single root, in real-life settings a complex XML document may have two or more roots, e.g., one for comments and one for the main body

**Solution:** G is false: always one root

H. A limitation of XML is that although it expresses trees it cannot express complex data structures such as undirected, weighted graphs

**Solution:** H is false: XML can be used to express any data structure

I. The snippet

```
<!-- <anElement/> -->
```

means that a node for element anElement is placed as a child of the given comment node

**Solution:** I is false:

J. An element node can enclose attribute nodes as well as comment nodes, but an attribute node cannot enclose a comment node

**Solution:** J is true:

K. To represent information about a directed graph, it is best to specify an element for each vertex $V$ that takes an attribute for $V$'s identifier and another attribute containing a string of the identifiers of the target vertices of the edges that emanate from $V$

**Solution:** K is false: we shouldn't have to parse an attribute value to determine the vertices to which a given vertex is connected

L. Using XML syntax for the XSLT language was an excellent design decision because it improved readability of XSLT stylesheets

**Solution:** L is false:

M. XML InfoSet specifies that comments may occur within attributes though most processors don't implement this feature

**Solution:** M is false:

N. By using a standard metadata language, programmers can avoid having to agree ahead of time on the specifics of the metadata they assign to the documents they exchange

**Solution:** N is true: if programmers agree on the language, they and their programs can compute the meaning associated with a specific document as needed

O. An element $X$ may enclose two or more text nodes as long as any two consecutive text nodes are separated by other nodes, such as the subelements of $X$

**Solution:** O is true:

P. Text within elements is a convenience but anything that text can represent we can represent using attributes

**Solution:** P is true:

3. (22 points) Of the following statements, identify all that hold about XPath.

   A. The XPath expression child::Song[@language] finds the attribute nodes named language that are children of the Song children of the context node

   **Solution:** A is false: it finds Song elements that are children of the context node and have an attribute named language

   B. Because the axis descendant is of principal type element, the XPath expression descendant::node() yields exactly the same nodes that descendant::element() yields

   **Solution:** B is false:

   C. Languages that are based on XPath derive part of their power from the ability to write loops in XPath using the * construct

   **Solution:** C is false: no loops in XPath

   D. For a context node that has one or more preceding siblings, the expression preceding::node()[1] finds the immediately preceding sibling

   **Solution:** D is true:

   E. The XPath descendant and ancestor axes are inverses of each other: each node is a descendant of each of its ancestors

   **Solution:** E is false: attributes are not

   F. In XPath, the descendant axis is the transitive closure of the child axis

   **Solution:** F is true:

   G. If an XPath expression E yields a node sequence consisting of exactly one element, then E[$-1$] = E[last()]

   **Solution:** G is false: only integer filters from $1$ to last() can produce a result; when there is an element, E[last()] produces an element

   H. For any XPath expression E, we have E[$-1$ or $-1$] = E[true()]

   **Solution:** H is true: the effective Boolean value of $-1$ is true

   I. For any XPath expression E, we have E[$-1 + -1$] = E[$-1 * -1$]

   **Solution:** I is false: in general, E[$-2$] $\neq$ E[1]

   J. For no XPath expression E, do we have E[$-1$ or $-1$] = E[$-1$]

**Solution:** J is false: when $E$ yields the empty node sequence, all further selections from it yield the empty node sequence

K. The self axis is defined for every element regardless of how deep it occurs in a tree

**Solution:** K is true:

4. (16 points) Of the following statements, identify all that hold about XQuery. (Below, Set and Pred are functions and $x and $v are variables.)

A. In XQuery, an easy way to increment a variable $x is to write let $x := $x + 1

**Solution:** A is false: no side effects in XQuery

B. Because XQuery does not allow side effects on variables, we are sometimes forced to create solutions of exponential complexity for problems that would otherwise take solutions of polynomial complexity. An example is the problem of calculating the first $n$ Fibonacci numbers. Fortunately, those problems are not real business problems

**Solution:** B is false:

C. Any XQuery query that produces a result must include exactly one return

**Solution:** C is false: only each FLWOR expression must include exactly one return

D. In XQuery, the collector variable paradigm works for a function that is recursive but not for two functions that are mutually recursive

**Solution:** D is false:

E. Given for $x in Sequence . . . , where Sequence is some expression whose evaluation terminates, if the body of the for terminates for each possible binding of $x, the entire for expression must terminate

**Solution:** E is true:

F. In XQuery, we can compute all the axes of XPath using no axes other than parent and child

**Solution:** F is false: we can compute almost all of the axes; however, the attribute axis is not captured by any XQuery query involving just parent and child

G. In for $x in Sequence . . . , the expression Sequence may seem to use $x but may *not* refer to the same $x that is declared in the current for construct

**Solution:** G is true: Sequence should evaluate to the node sequence from which the values taken by $x are drawn and would be meaningless if it referred to the same $x

H. In for $x in Sequence return Result, the expression Result may seem to use $x but may *not* refer to the same $x that is declared in the current for construct

**Solution:** H is false: Result should evaluate to the outputs for each binding of $x and not only may but usually must refer to the same $x (so as to access its binding value)

5. (16 points) Of the following statements, identify all that hold about XSLT.

   A. XSLT places many important expressions inside strings as values of attributes

   **Solution:** A is true: Much of the important work of XSLT is based on XPath expressions placed as values for the attributes match, test, and select

   B. In XSLT as studied in this course, XPath expressions occur only in three places: as values for the attributes match, test, and select

   **Solution:** B is true:

   C. Any XSLT stylesheet that does not explicitly use apply-templates is guaranteed to terminate

   **Solution:** C is true: by default, the application of templates would recurse down the tree and therefore terminate at nodes that have no children

   D. Any XSLT stylesheet that explicitly uses apply-templates on the context node's parent is guaranteed to terminate

   **Solution:** D is false: one could end up with a nonterminating execution where the parent invokes apply-templates on a child and the child on the parent

   E. It is possible to define an XSLT template that (i.e., the same template) handles both recursion going down the document tree and up the document tree

   **Solution:** E is true:

   F. In XSLT, we can generate element nodes whose names are not fixed a priori, but rather are determined programmatically at run time

   **Solution:** F is true:

   G. In XSLT, it is possible to have templates that match on the XPath expression / even though that expression corresponds to the unique root node that does not even occur within the main part of an XML document

   **Solution:** G is true:

   H. Any legal XPath expression, $X$, that evaluates to an element or set of elements may be used as a match pattern for an XSLT template

   **Solution:** H is true: