1. (14 points) Of the following statements, identify all that hold about e-business concepts.

   A. Autonomy and heterogeneity are two names for the same concept

   Solution: A is false: autonomy deals with action and heterogeneity with design and construction, so they are quite different

   B. Dealing with open systems is difficult because they require us to construct tightly integrated solutions

   Solution: B is false: open systems do not require integrated solutions; arms-length relationships are the best way to develop solutions fo open systems

   C. A TP monitor cannot necessarily ensure consistency among the resources that it controls

   Solution: C is false: to ensure consistency is the main function of a TP monitor; the resources a TP monitor controls would reside within the same administrative domain

   D. In open environments, coherence is a more reasonable goal than consistency

   Solution: D is true:

   E. The dynamism of an IT system refers to the freedom of system administrators to alter the configurations of the modules in the system

   Solution: E is true:

   F. In a nutshell, what makes open business environments difficult to deal with are sociopolitical and historical factors such as that different business partners are free to act as they please and their business modules may be implemented in diverse ways

   Solution: F is true:

   G. Some techniques that address closed environments may also be applied in open environments and can help improve robustness and productivity

   Solution: G is false:

2. (18 points) Of the following statements, identify all that hold about metadata and XML.

   A. To use information well requires capturing context, yet to reuse information well requires that we can capture its meaning in a context-free manner

   Solution: A is true: to reuse information presupposes that it holds some meaning even as the context of use changes

   B. An XML namespace is not a resource and therefore it is bad practice to try to give it a URI

   Solution: B is false: anything can be treated as a resource and in particular a namespace makes an excellent resource

C. If we give a URI to an XML namespace, the URI resolves at the schema where the namespace is defined

> **Solution:** C is false: a URI is just an identifier

D. Whereas in most beginner tutorials an XML document corresponds to exactly one tree, a complex XML document may correspond to two or more trees

> **Solution:** D is false: always one tree

E. XML documents that cannot be parsed are useful for capturing browser interfaces and configuration data

> **Solution:** E is false: XML documents that cannot be parsed are not XML documents, and are thus useless as XML

F. We sometimes cannot design XML documents that express important metadata for real-life situations, because XML doesn't allow attributes to have attributes

> **Solution:** F is false:

G. An element with no text, no attributes, and no subelements represents a null value

> **Solution:** G is false: we need the xsi:nil attribute

H. Specifying units within attribute values as in <bill amt='USD 10'/> is desirable because the metadata USD makes sure the value 10 won't be misinterpreted

> **Solution:** H is false: we shouldn't have to parse attribute values further; we should capture units via a separate attribute or instead use elements with appropriate subelements and attributes

I. Ultimately, any type of metadata can be useful only if the creators and readers of that metadata interpret it sufficiently similarly

> **Solution:** I is true: there must be sufficient agreement or effective communication would be impossible

3. (18 points) Of the following statements, identify all that hold about XPath.

A. The XPath expression child::Song[Sgr] finds the Sgr elements 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

B. The XPath expression child::node() would yield any of the nodes that child::element() yields

> **Solution:** B is true:

C. XPath supports what might be termed the *tags* view of an XML document

> **Solution:** C is false: it is the template view

D. XPath's preceding axis helps find nodes that precede the context node, such as its preceding siblings, parent, and other ancestors

> **Solution:** D is false: not its parent or other ancestors

E. The XPath child and parent axes are inverses of each other: each node is a child of its parent node

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

F. Every element in an XML document has a parent

> **Solution:** F is true: even the top element has the document root (the / node) as its parent

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

> **Solution:** G is true:

H. A node that has one or more children is always the parent of each of its children

> **Solution:** H is true:

I. A node is always the descendant of each of its ancestors

> **Solution:** I is false: attributes are not

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. The collector variable paradigm for programming works in XQuery

> **Solution:** A is true:

B. Any XQuery query that produces a result must include at least one for or let

> **Solution:** B is false: only each FLWOR expression must include at least one for or let

C. Any FLWOR query that produces a nonempty result must include exactly one where clause

> **Solution:** C is false: a where is optional

D. This course advocates using higher-level languages such as XQuery to extract information from XML documents

> **Solution:** D is true: yes, strongly!

E. XML query languages such as XQuery draw upon previous work on query languages for object-oriented databases

> **Solution:** E is true:

F. XQuery shows how to query XML documents without using XML syntax

> **Solution:** F is true:

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

> **Solution:** G is false: the expression Sequence is calculated before the for line is executed so if the calculation of the Sequence expression never terminates, the whole for expression won't terminate either

H. Given XQuery, a lot of XPath is redundant. For example, using no axes other than parent and child, we can write an XQuery function that would compute the *following siblings* of its argument element

> **Solution:** H is true:

5. (10 points) Of the following statements, identify all that hold about XSLT.

A. Using XSLT, we can easily modify an input XML document that obeys one schema to produce a document that obeys an entirely different schema

> **Solution:** A is false: the idea with XSLT is to produce a new document from an input, but there are no modifications

B. A simple way to ensure that your XSLT stylesheet will terminate on all inputs is to only use the child, following, following-sibling, descendant, and descendant-or-self axes in XPath expressions within the stylesheet

> **Solution:** B is false: can create an infinite loop through the descendant-or-self axis

C. The collector variable paradigm for programming works in XSLT

> **Solution:** C is true:

D. It is possible to have two templates in the same stylesheet such that one template matches Song[1] and the second template matches Song

> **Solution:** D is true:

E. Any legal XPath expression, X, may be used as a pattern on which an XSLT template may match, i.e., we can write <xsl:template match='X'>. . . </xsl:template>

> **Solution:** E is false: some XPath expressions are not valid patterns for matching templates; for example, the expressions '55' and '1+1'