1. (24 points) Identify all of the following statements that are true about the basics of services.

   A. If you know that two parties implement SOAP, then you can safely conclude they will interoperate at least in terms that SOAP addresses, namely, at the level of exchanging requests and responses

   **Solution:** A is false: successful interoperation depends on what version of SOAP, each of them implements; recall our discussion of Basic Profile 1.0, which seeks to ensure interoperability that merely knowing the parties use SOAP cannot do

   B. Programmers need to learn the SOAP schema so as to offer and use Web services.

   **Solution:** B is false: the schema is useful for tools; programmers need to work at higher levels of abstraction

   C. Every URI is a URL

   **Solution:** C is false: only URIs that follow the URL schemes are URLs

   D. Screen scraping reflects the recommended best practice for extracting information from a web site

   **Solution:** D is false: screen scraping is the oldest way to extract information from a web site and far from recommended practice because it couples the information extracted with the ad hoc and changing structure of a web site's pages

   E. UDDI not only provides a registry but also provides hosting for services that a consumer can discover and execute on the UDDI engine

   **Solution:** E is false: UDDI is only a registry; there is no notion there of hosting services

   F. UDDI provides a natural basis for trusting services

   **Solution:** F is false: trust is one of the challenges that UDDI does not address

   G. The definition of services we have adopted in this course is that services are objects accessed remotely

   **Solution:** G is false: our definition is inspired from real-life services and is based on the idea of services as being engaged; this is far removed from the idea of objects

   H. The classical “triangle” architecture for services defines a way for a service consumer to send a request to a service

   **Solution:** H is true: the architecture includes SOAP as a way for a service consumer to send a request to a service

   I. The classical “triangle” architecture for services defines three main roles: a service consumer, a service provider, and a service monitor

   **Solution:** I is false: there is a service registry but not a service monitor
J. WSDL provides a way to describe a service based on its input and output signatures, but largely ignores the functionality the service provides

Solution: J is true: it has no account of the functionality provided by a service

K. In important use cases, a service provider and consumer carry out conversations, which the basic Web services standards (SOAP, WSDL, UDDI) do not support

Solution: K is true:

L. A fundamental assumption of services is that though the participants may be autonomous, they are never selfish and will do always do right by others

Solution: L is false: there is no assumption that services are always well-behaved; that’s why, for instance, compliance remains a big challenge

2. (8 points) Consider a value map between the set of letters, \( a = \{A \ldots Z\} \) (sorted with A as lowest and Z as highest) and a set of integers, \( b = \{1 \ldots 4\} \). As usual, we consider the value map as two functions, \( m_{ab} \) and \( m_{ba} \). Let \( m_{ab} \) map \( \{A, B\} \) to 1; \( \{C, D\} \) to 2; \( \{E, F\} \) to 3; and map all the remaining letters \( \{G \ldots Z\} \) to 4.

The following are mutually independent problems based on the above setting.

A. \( m_{ab} \) is order preserving and total

Solution: A is true: by fact of being defined for all letters and in an order preserving manner

B. If \( m_{ba} \) is order preserving and total, then \( m_{ab} \) is necessarily consistently inverting

Solution: B is false: for example, let \( m_{ba} \) map 1 to C (and the rest don’t matter)

C. If \( m_{ba} \) maps any two numbers to the same letter, it cannot be consistently inverting

Solution: C is false: for example, let \( m_{ba} \) map all of the numbers \( \{1 \ldots 4\} \) to A; then, for any number \( n \), \( m_{ba}(n) = A \), \( m_{ab}(m_{ba}(n)) = 1 \), and \( m_{ba}(m_{ab}(m_{ba}(n))) = A \), thus ensuring consistent inversion

D. No matter what \( m_{ba} \) is, if it is total and if \( m_{ab} \) and \( m_{ba} \) are consistent inverses of each other, then \( m_{ba} \) must be order preserving

Solution: D is true: for \( m_{ab} \) and \( m_{ba} \) to be consistent inverses of each other, \( m_{ba} \) must map each number to a letter that maps to it; hence, it must be order preserving

3. (24 points) Identify all of the following statements that are true about knowledge modeling and RDF

A. When author in one conceptual model is creator in another conceptual model, this is an example of the process dimension of abstraction

Solution: A is false: it is an example of the structure dimension

B. That the expected shipping time is always positive is an example of the data dimension of abstraction
C. Whether you can cancel a booking or handle other contingencies is an example of the policy dimension of abstraction

Solution: C is false: it is part of the process dimension

D. A desirable property of a conceptualization is that it be modular

Solution: D is true:

E. A major advantage of a formal conceptualization is that it enables us to infer appropriate relationships without having to state each of the relationships explicitly

Solution: E is true: logic facilitates sound reasoning

F. We can apply an articulation axiom in both directions: to map concepts in one ontology map to concepts in another ontology, and in the reverse direction

Solution: F is true:

G. In RDF, we can capture a multiparty relationship via an association entity represented as a resource

Solution: G is true:

H. An RDF validator (i.e., reasoner) would flag conflicting assertions about the same resource

Solution: H is true:

I. In a well-formed RDF document, you cannot have two (different) assertions regarding the range of the same property

Solution: I is false: the reasoner infers the range to be the intersection of all the asserted ranges

J. A well-formed RDF Schema document is necessarily a well-formed XML document

Solution: J is false: RDF could be expressed in any notation, including the triples notation

K. A possible use for reification in RDF is to enable asserting comments about descriptions of services

Solution: K is true:

L. We can use RDF Schema to define a custom vocabulary that our instance documents can use to specify models of particular web resources

Solution: L is true:

4. (44 points) Identify all of the following statements that are true about RDF and OWL
A. OWL is nothing but a vocabulary for ontologies specified using RDF with constraints on how the terms in the vocabulary are to be interpreted

Solution: A is true: OWL is a vocabulary for ontologies built using RDF.

B. Anything you can represent in OWL DL, you can represent in RDF with the appropriate meaning conventions established.

Solution: B is true: OWL DL is simply a vocabulary defined using RDF and RDF Schema.

C. Anything you can represent in RDF, you can represent more precisely in OWL DL.

Solution: C is false: OWL constrains the interpretation of various constructs to give them a specific meaning; thus, RDF can be used with meaning conventions that may have no correlate in OWL DL; OWL DL lacks reification as well.

D. Using the OWL DL restriction construct, we can define a class based on a property (and incorporating suitable additional constraints).

Solution: D is true.

E. Let $F$ be a functional property in OWL DL with domain $D$ and range $R$. Then for each instance $d$ of $D$, there must be an instance $r$ of $R$ such that $d$ relates to $r$ through property $F$.

Solution: E is false: functionality imposes uniqueness, not totality (also known as existence in conceptual modeling).

F. Let $F$ be a functional property in OWL DL with domain $D$ and range $R$. Then for each instance $r$ of $R$, there must be an instance $d$ of $D$ such that $d$ relates to $r$ through property $F$.

Solution: F is false: functionality imposes uniqueness, not coverage of the entire range.

G. If an OWL DL object property is both symmetric and transitive, it is necessarily reflexive, meaning that each instance of the property’s domain relates to itself through the property.

Solution: G is false: as in elementary algebra, symmetry and transitivity do not entail reflexivity; you can imagine a class User, a symmetric, transitive property hasFriend, and yet have a particular instance of User who has no friends.

H. If you assert in OWL that both Class-A and Class-B are domains of the same property, then the OWL reasoner would conclude that Class-A is equivalent to Class-B.

Solution: H is false.

I. In OWL, it is possible to define a property whose domain and range are the same class.

Solution: I is true.

J. Because OWL has no variables, it is not possible to define a well-formed model in OWL that includes a cycle when rendered as a graph (with classes as vertices and properties as edges).
Solution: J is false:

K. We cannot correctly declare a resource to be an instance of two OWL classes in the same OWL document—for example, via ⟨Person rdf:ID='Pradeep'⟩ ⟨Student rdf:about='Pradeep'⟩

Solution: K is false: it is valid to make the above assertions; the OWL reasoner would then infer that Pradeep is both a Person and a Student, which is not a contradiction in an OWL model that reflects reality.

L. An OWL model naturally corresponds to a graph whose vertices correspond to resources and literals, and whose edges correspond to properties.

Solution: L is true:

M. In OWL owl:FunctionalProperty is actually a class, i.e., having rdf:type owl:Class

Solution: M is true:

N. If we assert that Medico is a superclass of Physician, then any property whose domain is Physician may potentially be asserted for an instance of Medico.

Solution: N is false: it may cause an error for an instance of Medico that is not an instance of Physician.

O. If we assert that Medico is a superclass of Physician, then any property whose range is Medico may potentially be asserted mapping an instance of its domain to an instance of Physician.

Solution: O is true: any instance of Physician is an instance of Medico.

P. Consider that we have defined two classes, Person and Parent and properties hasParent and parentOf in OWL; then, it is impossible to assert that Person is a subclass of Parent.

Solution: P is false.

Q. In OWL, it is acceptable to define two classes, Foo and Bar, and assert that Foo is a subclass of Bar and Bar is a subclass of Foo.

Solution: Q is true: When Foo and Bar are equal, each is a subclass of the other.

R. In OWL DL, subclass is a property whose domain is owl:Class.

Solution: R is true.

S. In OWL DL, we can assert that Class-A and Class-B are mutually exclusive subclasses of Class-C.

Solution: S is true: we can assert Class-A and Class-B are disjoint with each other and are subclasses of Class-C.
T. In OWL DL, we cannot assert that Class-A and Class-B are mutually exhaustive subclasses of Class-C

Solution: T is false: we can easily assert that, for example, by stating that Class-C is equivalent to the union of Class-A and Class-B.

U. An OWL DL data property cannot be symmetric

Solution: U is true: a data property has as its domain a class such as Person and as its range a set of literals such as integers; if it were symmetric, it would be like making an integer be a person; hence we cannot assert a data property to be symmetric.

V. An OWL DL data property can be transitive

Solution: V is false: a data property has as its domain a class such as Person and as its range a set of literals such as integers; if it were transitive, it would be like making an integer be a person; hence we cannot assert a data property to be transitive.

5. (14 points) Problems on dependencies and events (here \( \cap \) and \( \cup \) are delimiters)

A. Our formal syntax is such that the disjunction of two or more dot expressions involving different events, for example, \((e \cdot f \lor d \cdot \overline{f} \lor \overline{h} \cdot k)\), is a (well-formed) dependency.

Solution: A is true:

B. Our formal syntax is such that the disjunction of two events is itself an event.

Solution: B is false: an event is not a disjunction.

C. Our formal syntax is such that the conjunction of two events is itself an event.

Solution: C is false: an event is not a conjunction.

D. The requirement \("If e then f and if f then \overline{e}\) may be expressed as \(\tau\).

Solution: D is true: the English expression is formalized to \((\tau \lor f) \land (\overline{f} \lor \overline{\tau})\), which simplifies to \(\tau\).

E. The two dependencies, \(\overline{\tau} \lor e \cdot f\) and \(\overline{f} \lor f \cdot e\), cannot both be satisfied by the same schedule.

Solution: E is false: they are both satisfied as long as \(\tau\) and \(\overline{f}\) both occur—in any mutual order.

F. Given a dependency, we can allow any event whose residual with that dependency is nonzero.

Solution: F is true.

G. It is not possible to have a dependency \(D\), where \(D \neq \top\), such that \(D/e = D/\tau\).

Solution: G is false: consider any dependency involving events other than \(e\) or \(\tau\), e.g., the dependency \(f\).