1	2	3	4	5	6	7	total
20	20	20	50	10	10	20	150

Name:_____ UMBC ID: _____

UMBC CMSC 491/691 Final Exam
19 December 2016

Please write all of your answers on this exam, using the blank side of pages if you need more room. The exam is closed book/notes and has seven problems that add up to 100 points. You have the two hours to work on this exam. Good luck.

1. True/False (20 points)

T F XML was first defined as an extension of HTML for data. False

T F XML data model is based on a tree. True

T F Any constraints expressible in XML's document type definition (DTD) can also be expressed in XML Schema (XSD). True

T F Metadata is "data about data." True

T F RDF's data model is based on XML with extended data types. False

T F The N3 RDF serialization is like Turtle, but it also supports simple rules. True

T F Reification is used in RDF to make statements about triples. True

T F RDFS extends RDF by adding terms that allow default reasoning. False

T F Reasoning complexity in RDFS is decidable. True

T F It is not possible to state a logical contradiction in RDFS. True

T F One cannot make statements about properties in RDF or RDFS. False

T F OWL's data model is based on RDF, but introduces terms with new semantics. *True*

T F OWL-DL is an OWL profile based on Deontic Logic. False

T F OWL extends RDFS by allowing inherited properties to be overridden. False

T F An OWL class represents a set of individuals. True

T F Every OWL knowledge base can be serialized in the Turtle notation. True

T F Any inference that can be made by an OWL-DL reasoner can also be made using SWRL rules. False

T F The SPARQL language can only be used to query a triple store and cannot be used to delete or add triples. False

T F The Schema.org vocabulary was initially specified by a W3C standards committee. False

T F RDFS is adequate to define the terms in the Schema.org schemata. True

2. XML vs. RDF/OWL (20 points)

Describe the major differences between XML and the RDF/OWL as languages for representing information. Give examples of appropriate use cases for each language as well as their advantages and disadvantages.

XML is a language specified in SGML whose purpose is representing simple data structures in a standardized, serial format. It is intended as a way to serialize data for storage and later reuse and for exchanging data between programs. XML's underlying data model is that of a tree. Although there are several ways to give a schema for an XML format, these are independent of the XML data document and are mostly used to determine if an XML document complies with the schema or not. Thus, the schema does not allow one to infer additional information beyond what is encoded in the XML.

RDF is itself a data mode, i.e., an abstract framework for representing data. The data model is that of a "pure" graph, made up of nodes and edges.

RDF is extended by RDFS and OWL, which do not change the underlying data model but introduce new types of nodes and edges that can be sued to specify schema-level information have a well-defined meaning, e.g., rdfs:subClassOf, rdfs:domain, and owl:disjointWith. Thus, the schema and data information can be mixed in the same document.

The schema-level information associated with an RDF graph can be used to verify that the graph is well formed, but also typically allows on to infer additional data elements. For example, knowing that the :spouse relation is symmetric and that :p1 :spouse :p2 allows one to infer that :p2 :spouse :p1 must also be in the graph, enabling a system to add the edge if it is missing.

An RDF dataset can be serialized in any a number of standard ways, e.g., Turtle, N-triples and XML.

3. OWL Assumptions (20 points)

(a) OWL makes the **open-world assumption.** Explain what this this is and why it is appropriate for a knowledge representation language for the Web.

OWL is based on first order logic. In this system, as in many other logic frameworks, the **open-world assumption** is the assumption that the truth value of a statement may be true irrespective of whether or not it is *known* to be true. It is the opposite of the closed-world assumption, which holds that any statement that is true is also known to be true.

One of the motivations for using an open-world assumption for a Web-based knowledge system is that the Web is a very large, decentralized and open collection of pages. As such, not all pages with RDF information will be available at a given time, perhaps due to network nodes or collections being down. Another motivation was the desire to have a system grounded on a logic with a simple and well understood semantics.

(b) OWL also does not make the **unique name assumption**. Explain what this is and why not making it is a reasonable choice for a knowledge representation language for the Web

The unique name assumption is a simplifying assumption made in some ontology languages and description logics. In logics with the unique name assumption, different names always refer to different entities in the world.

The standard ontology language OWL does not make this assumption, but provides explicit constructs to express whether two nodes with the similar or even the same properties denote the same or distinct entities, owl:sameAs and owl:differentFrom.

# 4. Graph ⇔ Turtle (50 points)

Consider the RDF graph on the right.

(a) Serialize the graph (without any inferred nodes or edges) using Turtle. You need not show prefix definitions. A Turtle example is shown below the graph. (20)

(b) Add any nodes and/or edges that can be inferred to the graph on the right and also serialize the inferred nodes/edges in Turtle. (10)

(c) Draw a small star on the nodes that represent instances, as opposed to classes. (10)

(d) Write appropriate domain & range assertions for properties :wrote, :year and :title in Turtle (10)
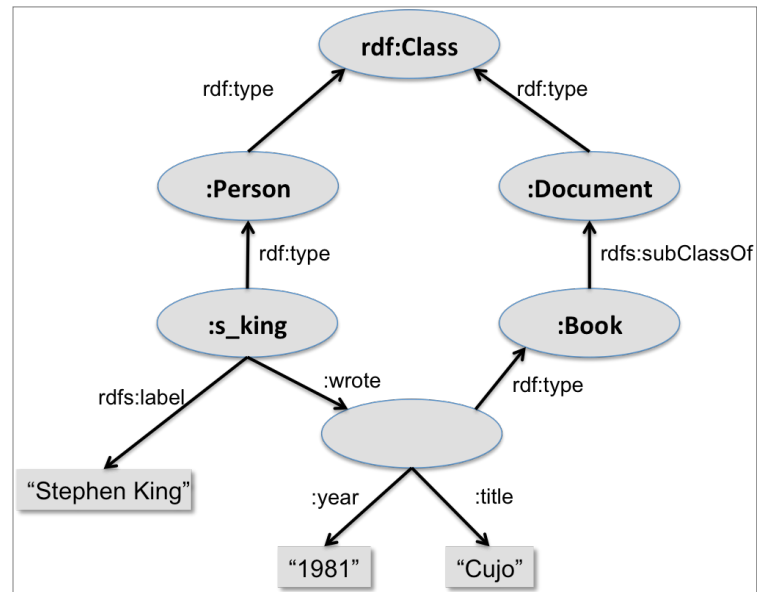

```
:trump a :Person;
    foaf:knows :pense, ivanka;
    :age "70"^^xsd:integer.
```

(a)
```
:Person a rdfs:Class.
:Document a rdfs:Class.
:Book a rdfs:Class; rdfs:subClassOf :Document.
:s_king a :Person;
    rdfs:label "Stephen King";
    :wrote _:bn01.
_:bn01 a :Book; :year ":1981"; :title "Cujo".
```

(b)   …modify graph to add a line from the blank node to the :Document node and label it with rdf:type …

```
_:bn01 a :Document
```

(c) the instances are the blank node and the node :s_king.

(d)
```
:wrote rdfs:domain :Person; rdfs:range :Document.
:year rdfs:Document; rdfs:range rdf:Literal.
:title rdfs:Document; rdfs:range rdf:Literal.
```

5. On blank nodes (10 points)

One of the nodes in the graph in the previous question is blank, i.e., it has no identifier associated with it. Explain the significance of and motivation for blank nodes in RDF/OWL.

In RDF, a blank node is a node in an RDF graph representing a resource for which a URI or literal is not given. The resource represented by a blank node is also called an anonymous resource. According to the RDF standard a blank node can only be used as subject or object of an RDF triple.

In a serialization, a blank node can be given a name, but it must start with the prefix _/ For example, _:person13 used as a node identifier it a blank node.

Blank nodes are like "local variables" in that they can not be refered to external to the local graph.

Blank nodes are useful when we know than an individual must exist, but don't know its identity. For example, if we know that :mary is the grandmother of :john, we can infer that there is a person _:x that is a child of Mary and a parent of John.

6. Disjoint classes (10 points)

We can assert that two classes are disjoint in OWL using the *owl:disjointWith* property, e.g., *:Man owl:disjointWith :Woman*. There are other ways to specify that two classes are disjoint without using this property. In Turtle, write two different ways to assert that :Man and :Woman are disjoint without using *owl:disjointWith*. (Hint: consider using *owl:ComplmentOf* and *owl:Nothing* along with other OWL/RFS constructors).

One way is to say that the intersection of: Man and: Woman is equivalent to owl:Nothing (the predefined class with no individuals in it). In Turtle, we first have to define a subject (i.e., a node) to represent the intersection and then say it's the equivalent to owl:Nothing:

```
:ManAndWoman owl:intersectionOf (:Man :Woman) .
owl:Nothing  owl:equivalentClass :ManAndWoman .
```

A second way is to say that :Woman is a subclass of the complement of :Man. In Turtle, we have to first define a subject (i.e., a node) to represent the complement of :Man, and then say :Woam is a subclass of this.

```
:NotMan owl:complementOf :Man .
:Woman rdfs:subClassOf :NotMan .
```

7. Embedding structured data in Web pages (20 Points)

Briefly describe the four major approaches in current use for embedding structured content in Web pages (i.e., Microformats, Microdata, RDFa and JSON-LD). Give at least one advantage and one disadvantage for each.

We didn't talk about **Microformats** this semester because while there are still a lot og web pages that have some microformat markup, it's not being used for new pages much.

Microdata is the native format developed for use with schema.org. It uses HTML tag attributes to encode markup, which is all assumed to refer to schema.org classes and properties. Its chief advantage is simplicity obtained by only allowing schema.org terms. One disadvantage is that it is not possible to use classes or properties from other namespaces/ontologies, like foaf or DBpedia. Another disadvantage is that schema.org is roughly like RDFS and does not have features that OWL adds that support more reasoning, such as being able to declare that a property is symmetric (e.g., spouse).

RDFa also uses HTML tag attributes to encode markup. Its big advantages are that the markup is not restricted to schma.org terms and can include terms from multiple namespaces/ontologies, including new ones that you invent. Since its based on RDF, all RDFS and OWL terms and features can be used. A disadvantage is that using it can be much more complicated than using Microdata, especially if you take advantage of the more complicated features or OWL. Using vocabulary other than schema.org will also not be understood by search engines.

JSON-LD is an approach in which the markup is not mixed in with the HTML, but segregated in (typically one) JSON object. An advantage of this is that it can be easier to generate the markup as a block and add it to the HTML content than it is to distribute the markup throughout the content. A disadvantage is that this introduces a possible problem of redundancy, where the same string (e.g., an address) needs to appear in both the html and JSON content. The burdon of ensuring that these are always the same may fall to the developer.