1	2	3	4	5	total
40	20	30	60	50	200

Name:_____ UMBC ID: _____

Knowledge Graphs (CMSC491/691) Midterm, 2018-10-22

Write your answers on this exam, using the blank side of pages if needed. You may only use notes on a single page of paper. You have 75 minutes to work on this exam. Good luck.

1. True/False (40 points)

T F The original HTML version was based on an early XML specification. F

T F XML's tree data model prevents it from representing graph structures. F

T F The values of an XML element's attributes must be strings. T

T F Any XML constraints expressible in XML Schema (XSD) can also be expressed in XML's Document Type Definition (DTD). F

T F An XML element cannot have two attributes with the same name. T

T F The nesting of XML elements has no standard meaning. T

T F XML's style sheet mechanism is intended to make XML data more human-readable. F

T F The order of triples in an RDF serialization carries no meaning. T

T F The subject of an RDF triple cannot be a string. T

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

T F RDF's syntax means it cannot be used to represent relations involving three entities. F

T F RDF's reification mechanism allows one to make statements about triples. T

T F An rdfs:subClassOf relation cannot hold between a node and itself. F

T F OWL uses, and does not extend, RDF's syntax. *T*

T F An OWL-DL class represents a set of individuals. T

T F An owl:inverseOf property cannot hold between an owl:ObjectProperty and itself. F

T F The owl:Nothing class is an rdfs:subClassOf the owl:Thing class. T

T F No owl property can be both an owl:functionalProperty and owl:inverseFunctionalProperty. F

T F An owl:datatypeProperty cannot have an inverse property. T

T F Owl's profiles (e.g., OWL-L, OWL-RL) restrict the features available in owl-full to make reasoning more manageable. T

2. owl:equivalentClass in RDFS (20)

Two OWL classes :C1 and :C2 are equivalent if every member of :C1 is necessarily a member of :C2 and vice versa. OWL provides a property to specify that two classes are equivalent, *owl:equivalentClass*. However, we could also specify that the two classes are equivalent just using RDFS vocabulary terms.

(a) Give an example in Turtle that makes using only RDF and RDFS vocabulary that asserts that classes :C1 and :C2 are equivalent. (10)

:C1 rdfs:subClassOf :C2 .
:C2 rdfs:subClassOf :C1 .

(a) Explain why this works (10)

If :C1 is a subclass of :C2, then every instance of :C1 is also an instance of :C2. Similarly, If :C2 is a subclass of :C1, then every instance of :C2 is also an instance of :C1. Thus, the two classes have the same instances and are therefore equivalent.

3. OWL properties (30 points)

(a) Briefly describe what it means for a property :P to be an *owl:functionalProperty* and give an example of a real-world property that could reasonably be modeled as an *owl:functionalProperty*. (10)

An owl FunctionalProperty is a property that can only have one (unique) value per object. Both object properties and datatype properties can be declared as "functional".

He property hasSSN (for social security number) might be modeled as a functional property for people in the U.S., since in any person should only have one SSN.

hasMother and hasFather (for biological parents) are other examples of functional properties, since a person only has one biological mother and father.

(b) Briefly describe what it means for a property :P to be an *owl:inverseFunctionalProperty* and give an example of a real-world property that could reasonably be modeled as one. (10)

An owl InverseFunctionalProperty is a property that has a unique value for each object that has that property. Both object properties and datatype properties can be declared as "inverse functional".

He property hasSSN (for social security number) might be modeled as an inverse functional property for people in the U.S., since a given SSN should only be assigned to one (living) person

hasMother and hasFather (for biological parents) are not inverse functional, since a person might be the mother of more than one child.

hasMobileNumber is an example that might be modelled as inverseFunctional (each mobile number is associated with just one person) but not functional (a person can have more than one mobile number)

(c) Is It possible for a property to be both an *owl:functionalProperty* and an *owl:inverseFunctionalProperty*? Briefly explain your answer.

Yes. The hasSSN example above is such a case.

4. Inference in OWL (60 points)

Review the facts serialized in Turtle in the figure. For parts b through f, assume you start with the facts in the KG and add the new triples to it. Circle the bullet next to your answer and, if it's the last one, enter the new facts in the space below.

```
:hasPet a owl:ObjectProperty;
   rdfs:domain :Person,
   rdfs:range :Animal .

:Person a owl:Class .
:Animal a owl:Class .
:Dog a owl:Class .
:Cat a owl:Class; owl:disjointWith :Dog .

# CatLover ≡ Person and (hasPet only Cat)
:CatLover a owl:Class ;
 owl:equivalentClass
   [ owl:intersectionOf (
      :Person
      [ a owl:Restriction;
         owl:onProperty :hasPet;
         owl:allValuesFrom :Cat ] )
   ].
:p1 a owl:NamedIndividual .
:p2 a owl:NamedIndividual .
:john  a owl:NamedIndividual .
:mary  a owl:NamedIndividual .
```

(a) What facts, if any can be inferred from this initial KG?

:Dog owl:disjointWith :Cat .

(b) If we add (**:john :hasPet :p1.**), what happens:

- A contradiction is discovered
- No new facts are inferred
- **The following new facts are inferred:**

:john a :Person .
:p1 a :Animal .

(c) If we add (**:john a :CatLover.**) what happens:

- A contradiction is discovered
- **No new facts are inferred**
- The following new facts are inferred:

(d) If we add (**:john a :CatLover; :hasPet :p1.**) what happens:

- A contradiction is discovered
- No new facts are inferred
- **The following new facts are inferred:**

:john a :Person .
:p1 a :Animal .
:p1 a :Cat .

(e) If we add (**:mary :hasPet :mary.**) what happens?

- A contradiction is discovered
- No new facts are inferred
- **The following new facts are inferred:**

:mary a :Person .
:mary a :Animal .

(f) If we add (**:mary a :CatLover; :hasPet p2. :p2 a :Dog.**) what happens?

- **A contradiction is discovered**
- No new facts are inferred
- The following new facts are inferred:

A reasoner will infer the following facts.

:mary a :Person .
:p2 a :Animal .
:p2 a :Cat .
:p2 a :Dog .

Since :Cat and :dog are disjoint constants, :p2 cannot be both. Thus a reasoner will not add any new facts.

5. Graph ⇔ Turtle (50 points)

Consider the graph on the right.

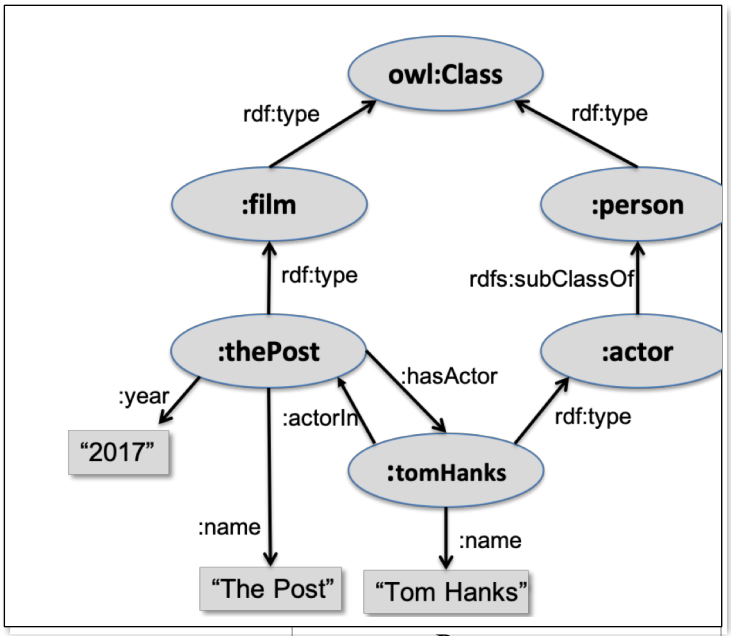
(a) Serialize the graph (without any inferred nodes or edges) using Turtle. You need not show prefix definitions. A Turtle example is shown below the graph. (20)

(b) Add 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 :year, :name, :hasActor and :actorIn in Turtle (10)

(e) Show assertions(s) to make properties :hasActor and :actorIn be inverses of one another.




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

(a)

```
:film a owl:Class.
:person a owl:Class.
:actor a owl:Class; rdfs:subClassOf :person.
:tomHanks a :actor;
    :name "Tom Hanks";
    :actorIn :thePost.
thePost a :film; :year "2017";
        :name "The Post"; :hasActor :tomHanks.
```

(b) …modify graph adding line from :tomHanks to the :Person node and label it with rdf:type …

```
tomHanks a :Person.
```

(c) the instances are the nodes :thePost and :tomHanks

(d)

```
:hasActor rdfs:domain :film; rdfs:range :actor.
:actorIn rdfs:domain :actor; rdfs:range :film.
:year rdfs:domain :film; rdfs:range rdf:Literal.
:name rdfs:domain owl:Thing rdfs:range rdf:Literal.
```

(e) :hasActor owl:inverseOf :actorIn