6.898 – Homework No. 4

By next class (Tuesday, October 12), all students should have done the following:

1. Use RDFS and OWL to express the following (assume a base URI of <http://www.example.com/ontology/life#>) in a valid RDF serialization:

"Some things are plants, others are animals, fungi, bacteria, archaea, or protists. Any Living Thing must be one of these, and no Living Thing may be more than one of these. Anything that is not a living thing is a non-living thing. All living things may be identified in part by the number of cells they have, but all must have at least one. Bacteria, Archaea, and Protists have exactly one cell.

Animals, fungi, plants, and protists are all eukaryotes. Things that are eukaryotes have a cell nucleus. Non-eukaryotes do not.

All living things belong to a species. Species may have many names, but are uniquely identified by no more than one scientific name (which is itself a name). Species also have no more than one date that they were first described and no more than one person (a foaf:Person) who first described them.

Some animals are Tailed Animals. Some animals are Legged Animals. Some animals are Legged Tailed Animals (and thus both). Any Legged Animal has a positive integral number of legs."

The following is one way to express this:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

# Note that we can't use owl:allDifferent, because these are classes.
:Plant a owl:Class ;
	rdfs:subClassOf :LivingThing , :Eukaryote ;
:Animal a owl:Class ;
	rdfs:subClassOf :LivingThing , :Eukaryote ;
	owl:disjointWith :Plant .
:Fungus a owl:Class ;
	rdfs:subClassOf :LivingThing , :Eukaryote .
	owl:disjointWith :Plant , :Animal ;
:Bacterium a owl:Class ;
	rdfs:subClassOf :LivingThing , :OneCelledOrganism , :NonEukaryote ;
	owl:disjointWith :Plant , :Animal , :Fungus .
:Protist a owl:Class ;
	rdfs:subClassOf :LivingThing , :OneCelledOrganism , :Eukaryote ;
	owl:disjointWith :Plant , :Animal , :Fungus , :Bacterium .
:Archaeon a owl:Class ;
	rdfs:subClassOf :LivingThing , :OneCelledOrganism , :NonEukaryote ;
	owl:disjointWith :Plant , :Animal , :Fungus , :Bacterium , :Protist .
:LivingThing a owl:Class ;
	owl:unionOf ( :Plant :Animal :Fungus :Bacterium :Protist :Archaeon ) ;
	owl:subClassOf [ a owl:Restriction ;
		owl:onProperty :species ;
		owl:cardinality 1 ] .

:cellCount a owl:DatatypeProperty ;
	rdfs:domain :LivingThing ;
	# Trick question!  We can't use owl:cardinality or owl:hasValue!
	# We have to use xsd:positiveInteger!
	rdfs:range xsd:positiveInteger .
```

```
:OneCelledOrganism a owl:Restriction ;
        owl:onProperty :cellCount ;
        owl:hasValue 1 .

:Eukaryote a owl:Restriction ;
        owl:onProperty :hasNucleus ;
        owl:hasValue "true"^^xsd:boolean .

:NonEukaryote a owl:Restriction ;
        owl:onProperty :hasNucleus ;
        owl:hasValue "false"^^xsd:boolean .

:species a owl:ObjectProperty ;
        rdfs:domain :LivingThing ;
        rdfs:range :Species .

:Species a owl:Class ;
        rdfs:subClassOf [ a owl:Restriction ;
                owl:onProperty :scientificName ;
                owl:maxCardinality 1 ] ,
                [ a owl:Restriction ;
                owl:onProperty :descriptionDate ;
                owl:maxCardinality 1 ] ,
                [ a owl:Restriction ;
                owl:onProperty :describedBy ;
                owl:maxCardinality 1 ] .

:name a rdf:Property ;
        rdfs:domain :Species .

:scientificName a owl:FunctionalProperty ;
        rdfs:subPropertyOf :name .

:descriptionDate a owl:DatatypeProperty ;
        rdfs:domain :Species ;
        rdfs:range xsd:date .

:describedBy a owl:ObjectProperty ;
        rdfs:domain :Species ;
        rdfs:range foaf:Person .

:TailedAnimal a owl:Class ;
        rdfs:subClassOf :Animal .

:LeggedAnimal a owl:Class ;
        rdfs:subClassOf :Animal ,
                [ a owl:Restriction ;
                owl:onProperty :legCount ;
                owl:cardinality 1 ] ,
                [ a owl:Restriction ;
                owl:onProperty :legCount ;
                owl:allValuesFrom xsd:positiveInteger ] .
:legCount a rdf:Property .

:TailedLeggedAnimal a owl:Class ;
        rdfs:subClassOf :TailedAnimal , :LeggedAnimal .
```

2. Write and execute SPARQL queries to do the following. Include the results of running the queries as a table (make sure to label your columns!). You may wish to use a SPARQL query viewer like that at http://ccgi.arutherford.plus.com/website/flex/dbPedia/sparqlQueryViewer/ to formulate and execute your queries (remember to change the endpoint appropriately!)

Query results, may vary. Working queries themselves are provided however.

a) Build an RDF graph that describes any actor in the movie "Avatar" as a member of a class <http://www.example.com/ont#AvatarActor>. This RDF graph should include the names of the actors by using the foaf:name predicate. Use the Linked Movie Database (http://www.linkedmdb.org/) to construct this query. You may wish to look at some of the links under "Start Exploring" (especially "actor" and "film") on that page to get an idea of the predicates you may want to use.

```
PREFIX movie: <http://data.linkedmdb.org/resource/movie/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

CONSTRUCT { ?x a <http://www.example.com/ont#AvatarActor> ; foaf:name ?name . }
WHERE {
        ?movie movie:actor ?x ;
                rdfs:label "Avatar" .
        ?x movie:actor_name ?name .
}
```

b) Get the name of all artists and their Icelandic-language albums which have been released in both Japan and the US. Also get the artist's birth and death dates (if they exist). Use DBTune's MusicBrainz database (http://dbtune.org/musicbrainz/). You may wish to look at some of the example links at the top of the page (especially "album", "artist", "release", and "language") to get an idea of the predicates you may want to use.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX bio: <http://purl.org/vocab/bio/0.1/>
PREFIX mo: <http://purl.org/ontology/mo/>

SELECT ?artist ?birth ?death ?album WHERE {
?x rdfs:label ?artist .
?y foaf:maker ?x ;
        dc:language <http://dbtune.org/musicbrainz/resource/language/isl> ;
        rdfs:label ?album ;
        mo:release ?americanRelease , ?japaneseRelease .
?americanRelease mo:release_country <http://dbtune.org/musicbrainz/resource/country/US> .
?japaneseRelease mo:release_country <http://dbtune.org/musicbrainz/resource/country/JP> .
OPTIONAL { ?artist bio:event ?event . ?event a bio:Birth ; bio:date ?birth . }
OPTIONAL { ?artist bio:event ?event . ?event a bio:Death ; bio:date ?death . }
}
```

c) Get the URI, name, founding date, and stadium names of all Major League Baseball teams established before 1900 that are/were tenants of a stadium that opened on or after 2000 OR that were established on or after 1900 and are/were tenants of a stadium that opened before 2000. This information may be found on DBPedia (http://dbpedia.org/).

A good place to start investigating predicates would be http://dbpedia.org/resource/Chicago_Cubs. Try to find the properties that give the name and founding date of the team and which property links to a resource that describes Wrigley Field, their stadium (where you should be able to find its opening date). In order to limit your search to Major League Baseball teams, you'll also want to find out what class the Chicago Cubs belongs to that represents Major League Baseball teams.

Note that DBPedia is particularly "cluttered" in terms of alternate ways of phrasing the same fact (that may not actually be useful to you!). You'll probably need to check several teams and stadiums to get a good idea of the best predicates to match the most teams.

DBPedia's SPARQL endpoint is http://dbpedia.org/sparql.

This query will vary greatly depending on the properties chosen.

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbpprop: <http://dbpedia.org/property/>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>

SELECT ?uri ?name ?founding ?stadium WHERE {
?uri a <http://dbpedia.org/class/yago/MajorLeagueBaseballTeams> ;
        rdfs:label ?name ;
        dbpprop:established ?founding .
?stadiumuri dbpedia-owl:tenant ?uri ;
        rdfs:label ?stadium ;
        dbpprop:openingDate ?stadiumyear .
FILTER ((?founding < "1900"^^xsd:integer && ?stadiumyear >= "2000-01-01"^^xsd:date) ||
          (?founding >= "1900"^^xsd:integer && ?stadiumyear < "2000-01-01"^^xsd:date)) .
FILTER (langMatches(lang(?name), 'en') && langMatches(lang(?stadium), 'en')) .
}
```

3. What are these SPARQL queries trying to do?

a)

```
PREFIX money: <http://telegraphis.net/ontology/money/money#>
PREFIX gn: <http://www.geonames.org/ontology#>
PREFIX geographis: <http://telegraphis.net/ontology/geography/geography#>
CONSTRUCT { ?l money:currency ?c ; gn:name ?y . ?c money:name ?x . } WHERE { {
?c      money:name ?x ;
        money:minorName “cent”@en ;
        money:currencyOf ?l .
?l      gn:name ?y .
} UNION {
?c      money:name ?x ;
        money:minorName “penny”@en ;
        money:currencyOf ?l .
?l      gn:name ?y ;
        geographis:officialLanguage <http://www.lingvoj.org/lingvo/en> .
} }
```

It is trying to construct a graph containing the name of a currency, the name of some object, and a triple which connects the two where the currency of the object either has a minor name of “cent” in English or the minor name “penny” in English where the officialLanguage of the object is English (as represented by the Lingvoj URI).

b)

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
INSERT { ?p foaf:interest <http://dbpedia.org/resource/Linked_Data> . }
WHERE { {
?p      a foaf:Person ;
        foaf:interest <http://dig.csail.mit.edu/2010/Courses/6.898/> .
} UNION {
?g      foaf:interest <http://dig.csail.mit.edu/2010/Courses/6.898/> .
        foaf:member ?p .
} }
```

It is trying to add a triple which states that any foaf:Person who is interested in <http://dig.csail.mit.edu/2010/Courses/6.898/> or is a member of something which has that interest also has an interest in Linked Data.

c)

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
DESCRIBE ?p WHERE { {
?d      a foaf:PersonalProfileDocument ;
        foaf:maker ?p .
?p      a foaf:Person ;
        foaf:age ?x ;
        foaf:interest <http://dbpedia.org/resource/Linked_Data> .
?q      foaf:knows ?p .
<http://www.w3.org/People/Berners-Lee/card#i> foaf:knows ?q .
FILTER ( ?x < "30"^^xsd:int )
} UNION {
?d      a foaf:PersonalProfileDocument ;
        foaf:maker ?p .
?p      a foaf:Person ;
        foaf:age ?x ;
        foaf:interest <http://dbpedia.org/resource/Linked_Data> .
<http://www.w3.org/People/Berners-Lee/card#i> foaf:knows ?p .
FILTER ( ?x < "30"^^xsd:int )
} }
```

It is trying to describe all people under 30 who are interested in Linked Data, have made a foaf:PersonalProfileDocument, and who either Tim Berners-Lee knows directly, or knows through someone else.

In addition to the above tasks, we request that students provide another paragraph describing an idea for the Idea Bank. These ideas should be of a potential application for linked data.

Short Paper: Homework groups are responsible for providing a short one-page paper that investigates the impact of linked data on the enterprise market. You may consider packaged applications, tools, data suppliers, professional services or new business models

Backgrounder: Homework Group 4 is responsible for providing a backgrounder on Stephane Corlesquet and Drupal before next class.

Please remember to turn in all homework to the appropriate Stellar homework dropbox by 4:00PM Tuesday, October 12, except for the backgrounder, which should be submitted no later than 11:59PM Monday, October 11. Stellar drop boxes will be made for each of the individual components of this assignment.