Solutions of Test 1
Data Modeling (2018-11-10)

Group 1

a)

```
MATCH (:Movie {title:'Star Wars'})<-[:ACTED_IN]-(p:Person)-
[:ACTED_IN]->(:Movie {title:'Indiana Jones'})
RETURN p
```

b)

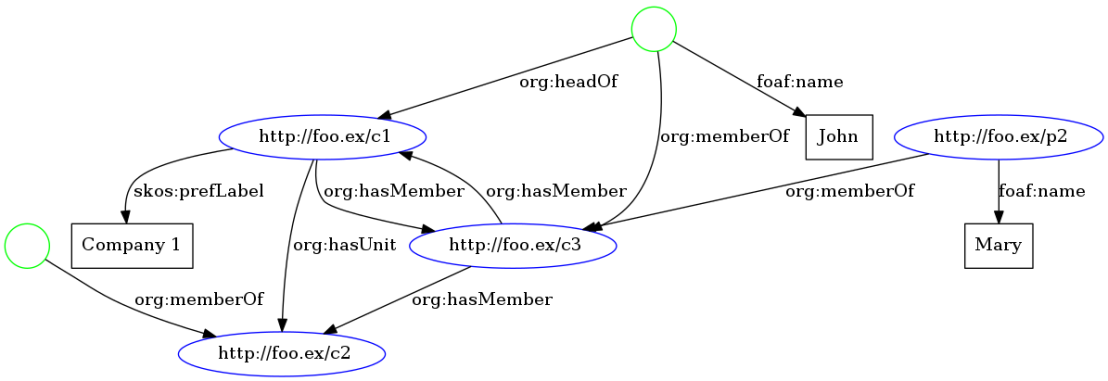
```
MATCH (m:Movie)
OPTIONAL
MATCH ()-[r:REVIEWED]->(m)
WITH m, r
RETURN m, collect(r.rating)
ORDER BY m.title

OR

MATCH (m:Movie)
OPTIONAL
MATCH ()-[r:REVIEWED]->(m)
WITH m, r
RETURN m, r.rating
ORDER BY m.title
```

c)

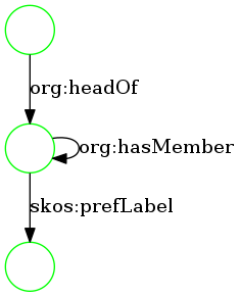
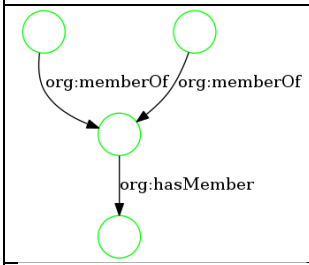
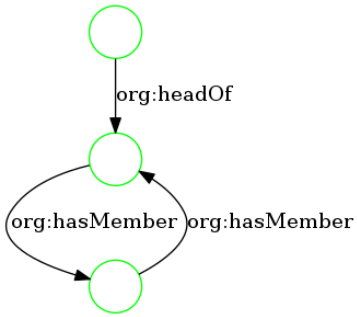
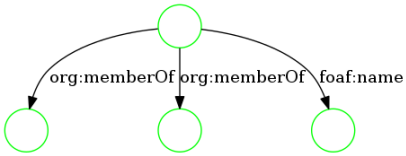
```
MATCH (p:Person)-[:WROTE]->(m:Movie)
WITH p, count(m) AS cnt_movies
WHERE cnt_movies > 3
RETURN p.name
```

Group 2

2a)



2b) By application of the interpolation lemma

	NO. Since < http://foo.ex/c1> has no loop to itself.
	YES. _:x is mapped into _:b1 _:y is mapped into < http://foo.ex/p2> _:z is mapped into < http://foo.ex/c3> _:w is mapped into < http://foo.ex/c1> or < http://foo.ex/c2>
	YES. _:x is mapped into _:b1 _:y is mapped into < http://foo.ex/c1> _:z is mapped into < http://foo.ex/c3>
	YES. _:x is mapped into _:b1 _:y is mapped into < http://foo.ex/c3> _:z is mapped into < http://foo.ex/c3> _:w is mapped into "John" also works with Mary.

2c) We need to infer the following four tuples (with URIs abbreviated)

_:x rdf:type foaf:Agent .
_:x org:memberOf _:y .
_:y rdf:type foaf:Agent .
_:y skos:prefLabel _:z .

The original triples are:

< http://foo.ex/c1> <http://www.w3.org/ns/org#hasMember> < http://foo.ex/c3> .
< http://foo.ex/c1> <http://www.w3.org/ns/org#hasUnit> < http://foo.ex/c2> .
< http://foo.ex/c1> <http://www.w3.org/2004/02/skos/core#prefLabel> "Company 1" .
< http://foo.ex/c3> <http://www.w3.org/ns/org#hasMember> < http://foo.ex/c1> .
< http://foo.ex/c3> <http://www.w3.org/ns/org#hasMember> < http://foo.ex/c2> .
< http://foo.ex/p2> <http://www.w3.org/ns/org#memberOf> < http://foo.ex/c3> .
< http://foo.ex/p2> <http://xmlns.com/foaf/0.1/name> "Mary" .
_:b1 <http://www.w3.org/ns/org#headOf> < http://foo.ex/c1> .
_:b1 <http://www.w3.org/ns/org#memberOf> < http://foo.ex/c3> .
_:b1 <http://xmlns.com/foaf/0.1/name> "John" .
_:b2 <http://www.w3.org/ns/org#memberOf> < http://foo.ex/c2> .

Including the schema triples and the graph triples (abbreviated):

1. org:memberOf rdfs:domain foaf:Agent .
2. org:memberOf rdfs:range org:Organization .
3. org:hasMember rdfs:domain org:Organization .
4. org:hasMember rdfs:range foaf:Agent.
5. org:headOf rdfs:subPropertyOf org:memberOf .
6. org:hasUnit rdfs:domain org:Organization .
7. org:hasUnit rdfs:range org:OrganizationalUnit .
8. org:OrganizationalUnit rdfs:subClassOf org:Organization .
9. org:Organization rdfs:subClassOf foaf:Agent .
10. :c1 org:hasMember :c3 .
11. :c1 org:hasUnit :c2 .
12. :c1 skos:prefLabel "Company 1" .
13. :c3 org:hasMember :c1 .
14. :c3 org:hasMember :c2 .
15. :p2 org:memberOf :c3 .
16. :p2 foaf:name "Mary" .
17. _:b1 org:headOf :c1 .
18. _:b1 org:memberOf :c3 .
19. _:b1 foaf:name "John" .
20. _:b2 org:memberOf :c2 .
21. _:b1 rdf:type foaf:Agent by rdfs2 on triples 1, 18
22. _:b1 org:memberOf :c1 by rdfs7 on triples 5, 17
23. _:c1 rdf:type org:Organization by rdfs2 on triples 3, 10 (there are several ways to conclude)
24. _:c1 rdf:type foaf:Agent by rdfs9 on triples 9, 23

We can now either apply interpolation lemma or apply se1/se2 to substitute _:b1 by _:x, :c1 by _:y, and "Company 1" by _:z to triples 21, 22, 24 and 12 obtaining the desired result.

Group 3

3a)

We have:

A = Bgp(?x ?foaf:name ?z) =

?x	?z
_:b1	"John"
:p2	"Mary"

B = Union(Bgp(?x org:memberOf ?y), Bgp(?y org:hasMember ?z)

?x	?y
_:b1	:c3
_:b2	:c2
:p2	:c3
:c3	:c1
:c1	:c3
:c2	:c3

Result = LeftJoin(A,B)

?x	?y	?z
_:b1	:c3	"John"
:p2	:c3	"Mary"

3b)

```
PREFIX …
SELECT ?o ?on ?p ?pn
WHERE
{ ?o a org:Organization .
  { { ?p org:memberOf ?o } UNION { ?o org:hasMember ?p } }
  OPTIONAL { ?o skos:prefLabel ?on }
  OPTIONAL { ?p foaf:name ?pn }
}
```

3c)

```
SELECT ?o ?on (COUNT(?p) AS ?num)
WHERE
{ ?o a org:Organization .
  ?o skos:prefLabel ?on .
  ?p org:memberOf ?o
}
GROUP BY ?o ?on
```

3d)

```
SELECT ?o ?s
WHERE
{ ?o a org:Organization .
  ?o (org:hasMember|^org:memberOf)+ ?s .
}
```