Name: **ID:**

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Total (max 31 points)
Points								

Knowledge Representation and Reasoning – Mod. 2

Mock Test 2022

RDF, RDFS, XSD & OWL Cheatsheet

- `rdf:type`
- `rdf:Statement`
- `rdf:subject`
- `rdf:predicate`
- `rdf:object`
- `rdfs:subClassOf`
- `rdfs:subPropertyOf`
- `rdfs:label`
- `rdfs:domain`
- `rdfs:range`
- `xsd:string`
- `xsd:integer`
- `xsd:decimal`
- `owl:equivalentClass`
- `owl:disjointWith`
- `owl:sameAs`
- `owl:differentFrom`
- `owl:Class`
- `owl:ObjectProperty`
- `owl:DatatypeProperty`
- `owl:NamedIndividual`
- `owl:Restriction`
- `owl:onProperty`
- `owl:allValuesFrom`
- `owl:someValuesFrom`
- `owl:cardinality`
- `owl:minCardinality`
- `owl:minQualifiedCardinality`
- `owl:maxCardinality`
- `owl:maxQualifiedCardinality`
- `owl:onClass`

Name: **ID:**

Question 1. RDF 5 P.

According to the semantics of *simple interpretations*:

1. Does the graph G_1 entail the graph G_2? Justify your answer (provide a valid blank node assignment or a counterexample)[1]. 2.5 P
2. Does the graph G_1 entail the graph G_3? Justify your answer (provide a valid blank node assignment or a counterexample). 2.5 P

G_1

```
ex:alice ex:parent ex:bob
ex:bob ex:parent ex:carol
```

G_2

```
_:x ex:parent _:y
_:z ex:parent _:x
```

G_3

```
_:x ex:parent ex:bob
ex:bob ex:parent _:x
```

Remark: prefix definitions are omitted; notation is Turtle.

Another example for Question 1 5 P.

Express the following using the Turtle syntax[2] and, where relevant, typed literals:

1. Alice knows Bob who is a person
2. Alice knows Mary, who is a Teacher and is 28 years old
3. Alice knows Bob's sister, who is 28 years old
4. Alice knows that Bob is 28 years old
5. Bob wants to prevent Alice from studying Law

```
@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 ex: <http://example.org/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
## continue here
```

[1] In a real exam, the hint written within parentheses may be omitted

[2] You can use prefixes, and, in particular `ex:` as a prefix

Question 1. RDF 5 P.

According to the semantics of *simple interpretations*:

1. Does the graph G_1 entail the graph G_2? Justify your answer (provide a valid blank node assignment or a counterexample)[1]. 2.5 P
2. Does the graph G_1 entail the graph G_3? Justify your answer (provide a valid blank node assignment or a counterexample). 2.5 P

G_1

```
ex:alice ex:parent ex:bob
ex:bob ex:parent ex:carol
```

G_2

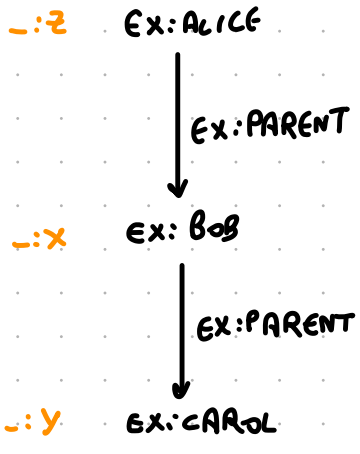
```
_:x ex:parent _:y
_:z ex:parent _:x
```

G_3

```
_:x ex:parent ex:bob
ex:bob ex:parent _:x
```

Remark: prefix definitions are omitted; notation is Turtle.

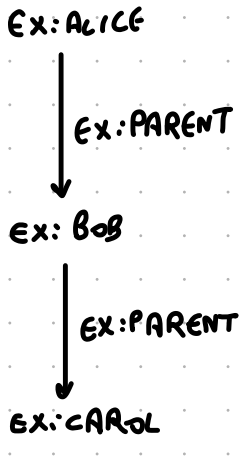
1)

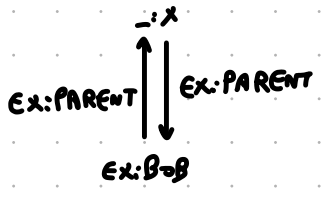


THE GRAPH G_1 ENTAILS THE GRAPH G_2

IF WE SUBSITUTE EX:ALICE TO THE BLANK NODE _:z, EX:BOB TO _:x AND EX:CAROL TO _:y

2)





NO, IF WE TRY ALL THE POSSIBLE SUBSITUTION OF THE BLANK NODE _:x WE WON'T REACH THE INTERPRETATION OF G_1

FROM EX. SESSION 1.

2. Does the RDF graph G_3 entail the RDF graphs G_4 and G_5? If not, give a counterexample.

RDF Graph G_3

```
ex:a  ex:p  _:b
_:c   ex:q  ex:d
```

RDF Graph G_4

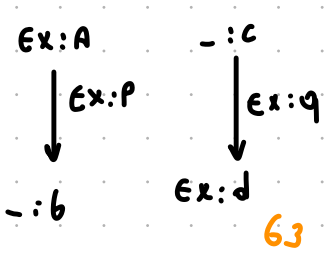
```
ex:a  ex:p  _:x
_:y   ex:q  ex:d
```

RDF Graph G_5

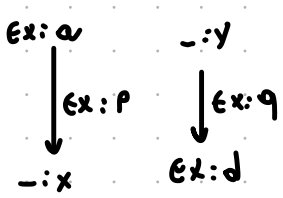
```
ex:a  ex:p  _:x
_:x   ex:q  ex:d
```

FARE TUTTE LE INTERPRETAZIONI POSSIBILI DEL GRAFICO DI PARENZA E VERIFICARE CHE POSSANO APPARIRE ANCHE NEL SECONDO GRAFICO

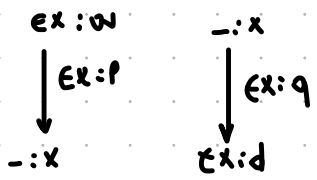
G_3



G_4



G_5



G_3 ALWAYS ENTAILS G_4

2 CASES

$_:b = _:c$

$_:b \neq _:c$

BOTH POSSIBLE IN G_4

G_3 NOT ENTAILS G_3

DOES NOT WORK IF $_:b \neq _:c$

Another example for Question 1 5 P.

Express the following using the Turtle syntax[2] and, where relevant, typed literals:

1. Alice knows Bob who is a person
2. Alice knows Mary, who is a Teacher and is 28 years old
3. Alice knows Bob's sister, who is 28 years old
4. Alice knows that Bob is 28 years old
5. Bob wants to prevent Alice from studying Law

```
@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 ex: <http://example.org/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
## continue here
```

1. EX:ALICE FOAF:KNOW EX:BOB. EX:BOB RDF:TYPE EX:PERSON.

2. EX:TEACHER RDF:TYPE OWL:CLASS. EX:PERSON RDF:TYPE OWL:CLASS.
   EX:TEACHER RDFS:SUBCLASSOF EX:PERSON.
   EX:ALICE FOAF:KNOW EX:MARY. EX:MARY RDF:TYPE EX:TEACHER;
   EX:YEARSOLD "28"^^XSD:INTEGER.

3. EX:ALICE FOAF:KNOW _:x. _:x FOAF:SISTEROF EX:BOB;
   EX:YEARSOLD "28"^^XSD:INTEGER.

4. EX:ALICE EX:KNOW [RDF:TYPE RDF:STATEMENT;
   RDF:SUBJECT EX:BOB;
   RDF:PREDICATE EX:YEARSOLD;
   RDF:OBJECT "28"^^XSD:INTEGER]

5. EX:BOB EX:WANT [RDF:TYPE RDF:STATEMENT;
   RDF:SUBJECT EX:BOB;
   RDF:PREDICATE EX:PREVENT
   RDF:OBJECT [RDF:TYPE RDF:STATEMENT; (OWL:NEGATIVEPROPERTYASSERTION)
   RDF:SUBJECT EX:ALICE;
   RDF:PREDICATE EX:STUDY;
   RDF:OBJECT EX:LAW]]

EX:SISTEROFA OWL:OBJECTPROPERTY;
RDFS:DOMAIN EX:PERSON;
RDFS:RANGE EX:PERSON.

EX:YEARSOLD RDF:TYPE OWL:OBJECTPROPERTY;
RDFS:DOMAIN EX:PERSON;
RDFS:RANGE XSD:INTEGER.

EX:KNOW RDF:TYPE OWL:OBJECTPROPERTY;
RDFS:DOMAN EX:PERSON;
RDFS:RANGE RDF:STATEMENT.

EX:WANT RDF:TYPE OWL:OBJECTPROPERTY;
RDFS:DOMAIN EX:PERSON;
RDFS:RANGE RDF:STATEMENT.
NEGATIVEPROPERTYASSERTION

EX:PREVENT RDF:TYPE OWL:OBJECTPROPERTY;
RDF:DOMAIN PERSON;
RDF:RANGE RDF:STATEMENT.

Name: **ID:**

Question 2. SPARQL

5 P.

Consider the following RDF graph:

```
@prefix : <http://example.org/> .
@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##> .

:A rdfs:subClassOf :B .
:B rdfs:subClassOf :C .
:p rdfs:domain :A ; rdfs:range :C .
:q rdfs:domain :B .

:a rdf:type :A . :b rdf:type :B . :c rdf:type :C .
:c :p :d . :d :q :e .
```

1. What are the results of the following queries? 2 P
   - `select * where { ?x a ?y }`
   - `select * where { ?x rdfs:subClassOf ?y } LIMIT 1`
2. Write the following queries and their results: 3 P
   - Retrieve all `?x` and `?y` such that `?x` and `?y` are related by `:p` or by `:q`.
   - Construct a graph containing the results of the following inference: if
     - `?c` is the domain of the property `?p`,
     - `?c` is a subclass of `?d`, and
     - `?x` and `?y` are related by `?p`

     then `?x` is an instance of `?d`

Question 2. SPARQL 5 P.

Consider the following RDF graph:

```
@prefix : <http://example.org/> .
@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##> .

:A rdfs:subClassOf :B .
:B rdfs:subClassOf :C .
:p rdfs:domain :A ; rdfs:range :C .
:q rdfs:domain :B .

:a rdf:type :A . :b rdf:type :B . :c rdf:type :C .
:c :p :d . :d :q :e .
```

1. What are the results of the following queries? 2 P
   - `select * where { ?x a ?y }`
   - `select * where { ?x rdfs:subClassOf ?y } LIMIT 1`
2. Write the following queries and their results: 3 P
   - Retrieve all `?x` and `?y` such that `?x` and `?y` are related by `:p` or by `:q`.
   - Construct a graph containing the results of the following inference: if
     - `?c` is the domain of the property `?p`,
     - `?c` is a subclass of `?d`, and
     - `?x` and `?y` are related by `?p`

     then `?x` is an instance of `?d`

AKA RDF:TYPE

1.1 (:a, :A); (:b, :B); (:c; :C)

1.2 (:A, :B)

2.1.
```
SELECT ?x ?y
WHERE { {?x :p ?y} UNION
        {?x :q ?y} }
```

2.2
```
CONSTRUCT {?x RDF:TYPE ?d}
WHERE { ?p RDF:DOMAIN ?c.
        ?c RDFS:SUBCLASSOF ?d.
        ?x ?p ?y}
```

Name: **ID:**

Question 3. RDFS 3 P.

Model (only) with RDFS syntax a schema that is compliant with the following constraints:

- Musicians and Bands are Music Artists;
- Musicians are associated with Bands;
- Musicians that sing in a band are associated with that band;
- Singles and Albums are Music Works;
- Singles are included in Albums;
- Singles have titles, and Albums have titles too;
- Titles are denoted with strings;
- Musicians and Bands make Singles and Albums;

and that guarantees the following inferences:

- If it is known that Paul McCartney (PM) makes Waterfalls, infer that PM is a Music Artist and Waterfalls is a Music Work;
- If a literal value v is a title of a Music Work, infer that it is a String;
- If it is known that PM sings in The Beatles, infer that PM is a Musician and The Beatles is a band;
- If it is known that Waterfalls in included in McCartney II, infer that Waterfalls is a Single and McCartney II is an Album.

You can skip the prefix section and use `ex:` as prefix for resources that are not classes or properties, and `onto:` as prefix for classes and properties (obvious exceptions apply to classes and properties defined in the `rdf`, `rfds`, and `xsd` vocabularies). Also you can draw a graph where nodes are classes and edges are properties in addition to writing the statements.

Question 3. RDFS 3 P.

Model (only) with RDFS syntax a schema that is compliant with the following constraints:

1 • Musicians and Bands are Music Artists;
2 • Musicians are associated with Bands;
3 • Musicians that sing in a band are associated with that band;
4 • Singles and Albums are Music Works;
5 • Singles are included in Albums;
6 • Singles have titles, and Albums have titles too;
7 • Titles are denoted with strings;
8 • Musicians and Bands make Singles and Albums;

and that guarantees the following inferences:

9 • If it is known that Paul McCartney (PM) makes Waterfalls, infer that PM is a Music Artist and Waterfalls is a Music Work;
10 • If a literal value v is a title of a Music Work, infer that it is a String;
11 • If it is known that PM sings in The Beatles, infer that PM is a Musician and The Beatles is a band;
12 • If it is known that Waterfalls in included in McCartney II, infer that Waterfalls is a Single and McCartney II is an Album.

You can skip the prefix section and use `ex:` as prefix for resources that are not classes or properties, and `onto:` as prefix for classes and properties (obvious exceptions apply to classes and properties defined in the `rdf`, `rfds`, and `xsd` vocabularies). Also you can draw a graph where nodes are classes and edges are properties in addition to writing the statements.

3° EX:SINGAN RDF:TYPE RDF:PROPERTY;
RDFS:DOMAIN EX:MUSICIAN;
RDFS:RANGE EX.:BANDS.
EX:SING IN RDFS: SUB PROPERTYOF EX: PLAYIN.

4° EX:SINGLE RDFS: SUBCLASS of EX:MUSICWORK.
6 X:ALBUM RDFS: SUBCLASS of EX:MUSIC WORK.

5° EX:INCLUDEDIN RDF:TYPE RDF: PREDICATE;
RDFS: DOMAIN EX;SINGLES;
RDFS: RANGE EX: ALBUM.

6-7 EX:HAVETITLES RDF: TYPE RDF:PREDICATE;
RDFS: DOMAIN EX:SINGLE;
RDFS: DOMAIN EX:ALBUM;
RDFS:RANGE EX:TITLE.

EX:TITLE RDF: TYPE XSD: STRING

- rdf:type
- rdf:Statement
- rdf:subject
- rdf:predicate
- rdf:object
- rdfs:subClassOf
- rdfs:subPropertyOf
- rdfs:label
- rdfs:domain
- rdfs:range

1° EX. MUSICIANS RDFS: SUBCLASSOF EX:MUSICARTIST.
EX: BANDS RDFS: SUBCLASSOF EX MUSICARTIST.

2° EX:ASSOCIATED RDF:TYPE RDF:PREDICATE.
EX:ASSOCIATED RDFS: DOMAIN EX:MUSICIAN
EX:ASSOCIATED RDFS:RANGE EX:BAND

EX:MUSICAN EX:ASSOCIATED EX:BAND.

8° EX: MAKE RDF:TYPE RDF: PREDICATE;
RDFS:DOMAIN EX: MUSICIARTIST;
RDFS:RANGE EX:MUSICWORK.

Name: **ID:**

Question 4. Modeling Knowledge with Description Logics (OWL) 5 P.

1. Express the following using the Description Logic syntax: 3 P
   a) Carol is a Math student who attends the Algebra course. Alice is a student who attends the Algebra and Algorithms courses. The latter is a CS course.
   b) Students are people who attend at least one course.
   c) Organized students are students who attend exactly 2 courses.
   d) Specify which classes (resp., properties) are subclasses (subproperties) of which; specify the domain and range of each property.
   e) Math students attend only Math courses, and CS students attend only CS courses.
2. What can be inferred in the Abox about Alice, Carol, and the Algebra and Algorithm courses, if anything (justify your answer)? 2 P

Question 4. Modeling Knowledge with Description Logics (OWL) 5 P.

1. Express the following using the Description Logic syntax: 3 P
   a) Carol is a Math student who attends the Algebra course. Alice is a student who attends the Algebra and Algorithms courses. The latter is a CS course.
   b) Students are people who attend at least one course.
   c) Organized students are students who attend exactly 2 courses.
   d) Specify which classes (resp., properties) are subclasses (subproperties) of which; specify the domain and range of each property.
   e) Math students attend only Math courses, and CS students attend only CS courses.
2. What can be inferred in the Abox about Alice, Carol, and the Algebra and Algorithm courses, if anything (justify your answer)? 2 P

1) :STUDENT RDF:TYPE :CLASS

RDF, RDFS, XSD & OWL Cheatsheet

- rdf:type
- rdf:Statement
- rdf:subject
- rdf:predicate
- rdf:object
- rdfs:subClassOf
- rdfs:subPropertyOf
- rdfs:label
- rdfs:domain
- rdfs:range
- xsd:string
- xsd:integer
- xsd:decimal
- owl:equivalentClass
- owl:disjointWith
- owl:sameAs
- owl:differentFrom
- owl:Class
- owl:ObjectProperty
- owl:DatatypeProperty
- owl:NamedIndividual
- owl:Restriction
- owl:onProperty
- owl:allValuesFrom
- owl:someValuesFrom
- owl:cardinality
- owl:minCardinality
- owl:minQualifiedCardinality
- owl:maxCardinality
- owl:maxQualifiedCardinality
- owl:onClass

4.1)

$MathStudent(Carol)$

$AttendCourse(Carol, Algebra)$

$Student(Alice)$

$AttendCourse(Alice, Algebra)$

$AttendCourse(Alice, Algorithm)$

$CourseOf(Algorithm, CS)$

$Student \equiv Person \sqcap \geq 1\, AttendCourse$

$OrganisedStudent \equiv Student \sqcap \geq 2\, AttendCourse \sqcap \leq 2\, AttendCourse$

$MathStudent \equiv Student \sqcap \forall AttendCourse.CourseOf(Course, Math)$

$CSStudent \equiv Student \sqcap \forall AttendCourse.CourseOf(Course, CS)$

4.2) Alice isn't neither a Math student nor a CS student.

Algebra is a course of Math.

Carol can't attend Algorithm.

Name: **ID:**

Question 5. Datalog 5 P.

1. Provide the most general unifier of the following pair of terms, if it exists. Justify your answer.

   `p(a, B, C, d, B)` `p(X, y, X, Z, X).`

2. Consider the following KB:

   ```
   r(a).  r(e).
   p(c).
   q(b).
   s(a, b).  s(d, b).  s(e, d).
   p(X) :- q(X), r(X).
   q(X) :- s(X, Y), q(Y).
   ```

   Show that KB $\vdash$ `p(e)` by proceeding bottom-up.

Question 5. Datalog 5 P.

1. Provide the most general unifier of the following pair of terms, if it exists. Justify your answer.

```
p(a, B, C, d, B)    p(X, y, X, Z, X).
```

{X/a, B/y, C/a, Z/d, NOT EXIST BECAUSE X/a, B/y AND X/B

WITH Y {X/a, B/Y, C/a, Z/d, Y/a}

2. Consider the following KB:

```
1) r(a).   2) r(e).
3) p(c).
4) q(b).
5) s(a, b).   6) s(d, b).   7) s(e, d).
8) p(X) :- q(X), r(X).
9) q(X) :- s(X, Y), q(Y).
```

Show that KB $\vdash$ `p(e)` by proceeding bottom-up.

C = { }

C = { r(a). r(e). p(c). q(b). s(a,b). s(d,b). s(e,d) }

SELECT CLAUSE N.9 AND SUBSTITUTE {X/d, Y/b}

C = { r(a). r(e). p(c). q(b). s(a,b). s(d,b). s(e,d). q(d). }

SELECT CLAUSE N.9 AND SUBSITUTE {X/e, Y/d}

C = { r(a). r(e). p(c). q(b). s(a,b). s(d,b). s(e,d). q(d). q(e). }

SELECT CLAUSE N.8 AND SUBSITUTE {X/e}

C = { r(a). r(e). p(c). q(b). s(a,b). s(d,b). s(e,d). q(d). q(e). P(e) }

Name: **ID:**

Question 6. Disjunctive and Nonmonotonic Reasoning in ASP — 5 P.

1. Consider the following program:

```
q v -r v p :- not p.
d :- not r.
-q.
```

   a) Explain why `p` cannot be in any answer set — 1.5 P

   b) Provide the answer sets of the program, if any. Justify your answer — 1.5 P

2. Write a program that implements the guess and check methodology in ASP according to the following specifications. — 2 P

   You have several socks of three colors and you want to organize them by assigning each sock to exactly one of three boxes:

   - Use the following predicates (and possibly more if you need):
     - `white_sock(X)`, `grey_sock(X)` and `black_sock(X)` if `X` is, respectively, a white, grey, or black sock.
     - `assign(X, Y)` means that the sock `X` is assigned to the box `Y`, where `Y` must be one of `white_box`, `grey_box` and `black_box`.
   - The organization must satisfy the following:
     - Grey socks can only be assigned to the grey box;
     - White socks can be assigned to the white box or the grey box;
     - Black socks can be assigned to the grey box or the black box;
     - You cannot put white and black socks in the same box.

   Additional question (not needed to get the full 2 points): How many possible ways to organize your socks (i.e., how many answer sets) are there if you have the following socks?

```
white_sock(w).  grey_sock(g).  black_sock(b).
```

Question 6. Disjunctive and Nonmonotonic Reasoning in ASP 5 P.

1. Consider the following program:

```
q v -r v p :- not p.
d :- not r.
-q.
```

a) Explain why p cannot be in any answer set 1.5 P

b) Provide the answer sets of the program, if any. Justify your answer 1.5 P

a) P CAN NOT BE SATISFIED IN ANY SET BECAUSE THE ONLY WAY TO HAVE IT IS TO SATISFY THE BODY OF THE FIRST CLAUSE THAT IS NOT P.

b) -q MUST ALWAYS BE IN THE ANSWER SET OF THE PROGRAM

```
q v -r v p :- not p.
d :- not r.
-q.
```
{-q, d}

```
q v -r v p :- not p.
d :- not r.
-q.
```
{-q, -r, p}

```
q v -r v p :- not p.
d :- not r.
-q.
```
{-q, d, -r}

```
q v -r v p :- not p.
d :- not r.
-q.
```
{-q, -r}

```
q v -r v p :- not p.
d :- not r.
-q.
```
{-q, p, d}

---

2. Write a program that implements the guess and check methodology in ASP according to the following specifications. 2 P

You have several socks of three colors and you want to organize them by assigning each sock to exactly one of three boxes:

- Use the following predicates (and possibly more if you need):
  - `white_sock(X)`, `grey_sock(X)` and `black_sock(X)` if `X` is, respectively, a white, grey, or black sock.
  - `assign(X, Y)` means that the sock `X` is assigned to the box `Y`, where `Y` must be one of `white_box`, `grey_box` and `black_box`.
- The organization must satisfy the following:
  - Grey socks can only be assigned to the grey box;
  - White socks can be assigned to the white box or the grey box;
  - Black socks can be assigned to the grey box or the black box;
  - You cannot put white and black socks in the same box.

Additional question (not needed to get the full 2 points): How many possible ways to organize your socks (i.e., how many answer sets) are there if you have the following socks?

```
white_sock(w).  grey_sock(g).  black_sock(b).
```

ASSIGN(X, GREY_BOX), ASSIGN(X, WHITE_BOX), ASSIGN(X, BLACK_SOCK) :- GREY_SOCK(X)

NO ASSIGN(X, WHITE_BOX) V ASSIGN(X, GREY_BOX) :- WHITE_SOCK(X)

SÍ ASSIGN(X, BLACK_BOX) :- WHITE_SOCK(X)

~~ASSIGN(X, GREY_BOX), ASSIGN(X, BLACK_BOX)~~ ASSIGN(X, WHITE_BOX) :- BLACK_SOCK(X)

:- ASSIGN(WHITE_SOCK, X), ASSIGN(BLACK_SOCK, X)

ASSIGN(Z, X) :- ASSIGN(K, X), BLACK_SOCK(Z), WHITE_SOCK(K)

ASSIGN(Z, X) :- ASSIGN(K, X), BLACK_SOCK(K), WHITE_SOCK(Z)

Name: **ID:**

Question 7. Open Question 3 P.

Explain the differences between *lexical ontologies*, *taxonomies*, and *axiomatic ontologies*.

```
q v -r v p :- not p.
d :- not r.
-q.
```