Knowledge Representation and Reasoning (KRR) Practice Questions

1. Theory Question

Question: Explain the difference between RDF and RDFS, and how RDFS extends RDF to provide additional semantic information. Include a brief explanation of how subclass relationships work in RDFS and provide an example.

Hint: - RDF allows for the representation of data using triples (Subject-Predicate-Object). - RDFS adds semantic meaning to RDF by introducing the concept of classes, properties, and hierarchical relationships. - Mention `rdfs:Class`, `rdfs:subClassOf`, and give an example of subclassing.

2. Turtle Code Question

Question: Write Turtle syntax to represent the following information:

- John is a `Person`.
- John works at a company called `TechCorp`.
- John has a colleague, Mary, who also works at `TechCorp`.
- John's position is `Software Engineer`, and Mary's position is `Product Manager`.

Hint: - Use meaningful URIs for entities (e.g., `ex:John`, `ex:TechCorp`). - Use properties to define relationships (e.g., `ex:worksAt`, `ex:colleagueOf`, `ex:position`).

3. Conversion Question: Graph to Code & Code to Graph

Part 1: Graph to Turtle Code

Question: Given the following graph:

- There is a class called `Animal`.
- There are two instances: `Lion` and `Elephant`.
- `Lion` is a subclass of `Animal`.
- `Elephant` is a subclass of `Animal`.
- `Lion` has the property `hasHabitat` with the value `Savannah`.
- `Elephant` has the property `hasHabitat` with the value `Forest`.

Write the corresponding Turtle code for this graph.

Hint: - Use the class `ex:Animal` and properties like `ex:hasHabitat`. - Define the subclass relationships using `rdfs:subClassOf`.

Part 2: Turtle Code to Graph

Question: Given the following Turtle code, draw the corresponding graph that represents the information:

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

ex:Car a ex:Vehicle.
ex:Car ex:hasColor "Red".
ex:Car ex:hasSpeed "120km/h".
ex:Driver ex:drives ex:Car.
```

In the graph, show the nodes and the relationships between them visually.

Hint: - The graph should include nodes for `ex:Car` and `ex:Driver`, with arrows showing properties like `hasColor`, `hasSpeed`, and `drives`. - You can use circles for entities and label the arrows with the relationships.