1. Ontology Definition:
   An ontology is defined as an explicit, formal specification of shared conceptualization. This additionally means that ontology provides a structured, clear and formally defined representation of knowledge within a particular domain, allowing shared understanding across applications and users.
   The concepts and relationships are expected to be clearly defined while also being represented in a logical machine-readable format. It is also expected to be represented in an structure agreed upon across stakeholders and this is crucial for interoperability in fields like the semantic web.
   Ref: Textbook, pages 1-17
2. Web Ontology Language (OWL):
   OWL 2 DL is a subset with computational completeness and is designed to allow for efficient reasoning It is used in applications where logical consistency is essential and is based on description logic SROIQ, therefore offering decidable reasoning.
   OWL 2 Full provides extensive expressiveness compatible with RDF but is undecidable, i.e; there is no complete reasoning support for OWL 2 Full
   OWL 2 also includes three profiles to facilitate tailored use in various domains by balancing complexity, reasoning and efficiency. They are as follows:
   - **OWL 2 EL**: This is optimized for large ontologies with numerous classes, supporting scalable reasoning
   - **OWL 2 QL**: This is suited for applications requiring efficient query answering, often used in cases related to databases
   - **OWL 2 RL**: This is designed for rule-based applications and help in balancing expressiveness and scalability
4. Family Ontology
   1) HappyPerson:
      DL: Happy Person ≡ (**hasChild** ⊓ ∃**hasChild**.HappyPerson ⊓ ∀**hasChild**.HappyPerson)
      Natural Language: A Happy person is defined as a person who has atleast one child, all of whom are also happy persons. Essentially, every child of a happy person must also be happy.
      JohnsChildren:
      DL: JohnsChildren ≡ ∃**hasParent**.{John}
      Natural Language: JohnsChildren is a class of individuals who have John as their parent
      NarcisticPerson:
      DL: NatcisticPerson ≡ **loves**.Self
      Natural Language: A narcissistic person is someone who loves themselves.
      Orphan:
      DL: Orphan ≡ (**inverse**(**hasChild**) ⊓ ∀**inverse**(**hasChild**).Dead)
      Natural Language: An Orphan is an individual whose parents are all deceased.
      Grandfather:
      DL: Grandfather ⊑ Man ⊓ Parent
      Natural Language: A grandfather is a person who is both a man and a parent.
   2) Teenager
      Natural Language: Teenage includes individuals who are between 12 and 19 years old, inclusive of the upper limits of the range, but not the lower limit.
   3) hasFather:
      DL: hasFather ⊑ hasParent ⊓ Male

Natural Language: The **hasFather** property describes an individual as the male parent of another individual. It is defined as the inverse of **hasChild** but constrained to male individuals only
hasGrandparent:
DL: hasGrandParent ≡ hasParent ∘ hasParent
Natural Language: The **hasGrandparent** property defines an individual who is the parent's parent of another individual. This relationship is established by chaining two **hasParent** relationships.
hasUncle:
DL: hasUncle ≡ hasParent ∘ hasBrother
Natural Language: The **hasUncle** property describes an individual as the parent's brother of an individual. This relationship is derived by combining the **hasParent** and **hasBrother** relationships.

4) According to the OWL 2 DL property and class assertions, we get that **John** is an individual who **has a wife named Mary** and **is 51 years old**. He is **identified as a father** and **has precisely three children** who are also **Parents**. Additionally, he **has exactly 5 children** in total. Of his children, he **has at most four who are also parents**, but **atleast two of his children are Parents**. John is recognized as being the **same individual as JohnBrown** but is explicitly stated to be a **different individual from Bill**.
5) Using the HermiT Reasoner, here are a few of the changes that came highlighted in yellow after reasoning:
    - MyBirthdayGuests (Class): **New instances were inferred** to this class. These instances were individuals, **namely Bill, John, JohnBrown, Mary and MaryBrown**. These individuals were referred because the MyBirthdayGuests class was initialized to be equivalent to the list {Bill, John, Mary} and additionally, the remaining classes inferred that John is the same as JohnBrown and that Mary is the same as MaryBrown.
    - hasParent (Object Property): New equivalence was inferred to hasParent that states that it is the inverse of **hasChild**, and **child** property.
    - Jack (Individual): A new type for Jack was inferred that states that **Jack is a ChildlessPerson**. Additionally, an object property assertion was inferred that stated that **Jack has a relative named Jack** who is himself, or that Jack has himself as his relative.

5.
1) As a typical European University, the minimal items needed to be kept track of are:
    - Courses
    - Degrees
    - Staff
    - Students
    - Subjects

   This ontology should keep track of how courses are structured, which individual teaches/learns which course and which individual works at the college

3) Competency Questions:
    - What are the available bachelor's and master's programs?
    - Which subjects are covered within a particular course?
    - Which staff members are teaching staff and which are non-teaching?

- Who are the staff teaching a particular course?

Additional Restrictions:

- Teaching Restriction: Specify that only teaching staff can teach courses
- Role Constraint: Teaching and Non-Teaching staff should be disjoint

Reasoning Requirement:

- Consistence Checking: Ensure consistency, especially for restrictions.
- Disjoint Class Assertions: Define and reason over disjoint classes such as between teaching and non-teaching staff.
- Instance Classification: Automatically classify individuals into relevant subclasses based on their properties