Answer to Question 1

OWL Functional syntax

```
Declaration(Class(:Vegetarian))
EquivalentClasses(:Vegetarian
                    ObjectIntersectionOf(
                        ObjectAllValuesFrom(:eat :Vegetable) :Person))
SubClassOf(:Vegetarian :Person)
DisjointClasses(:Vegetarian :NonVegetarian)
```

Manchester Syntax

```
Class: Vegetarian

   EquivalentTo:
      Person
       and (eat only Vegetable)

   SubClassOf:
      Person

   DisjointWith:
      NonVegetarian
```

Answer to Question 2

hasAunt:
Person(?x) ∧ hasParent(?x, ?y) ∧ hasSister(?y, ?z) → hasAunt(?x, ?z)
hasBrother
Person(?x) ∧ hasSibling(?x, ?y) ∧ Man(?y) → hasBrother(?x, ?y)
hasSister
Person(?x) ∧ hasSibling(?x, ?y) ∧ Woman(?y) → hasSister(?x, ?y)
hasSon
Person(?x) ∧ hasChild(?x, ?y) ∧ Man(?y) → hasSon(?x, ?y)
hasDaughter
Person(?x) ∧ hasChild(?x, ?y) ∧ Woman(?y) → hasDaughter(?x, ?y)
hasFather
Person(?x) ∧ hasParent(?x, ?y) ∧ Man(?y) → hasFather(?x, ?y)
hasNephew
Person(?x) ∧ hasSibling(?x, ?y) ∧ hasSon(?y, ?z) → hasNephew(?x, ?z)
hasNiece
Person(?x) ∧ hasSibling(?x, ?y) ∧ hasDaughter(?y, ?z) → hasNiece(?x,
hasUncle
Person(?x) ∧ hasParent(?x, ?y) ∧ hasBrother(?y, ?z) → hasUncle(?x, ?z)

hasSister is defined as a symmetric property

[hasSisterRule: (?x hasSister ?y)-> (?y hasSister ?x)]

If *p1* is a sub property of *p2,* if (*x, p1,y) then (x, p2, y)*
[subPropertyRule: (?p1 rdfs:subPropertyOf ?p2)(?x ?p1 ?y)->(?x, ?p2, ?y)]

hasAncestor is the transitive closure of *hasParent* Property

[hasParentRule: (?x hasParent ?y)->(?x hasAncestor ?y)]
[hasAncestorRule: (?x hasAncester ?y)(?y hasAncester ?z)->(?x hasAncester ?z)]

The instances of a subclass are also instances of its superclass

[hasSubclassRule: (?classA rdfs:subclassOf ?classB)(?classB rdfs:subclassOf ?classC)->(?classA rdfs:subclassOf ?classC)]
[instanceRule: (?x rdf:type ?classA)(?classA rdfs:subclassOf ?classB)->(?x rdf:type ?classB)]