1. **Translate** into **FOL**
    - Bowling balls are sporting equipment
        - $\forall x$ bowlingBall(x)→sportingEquipment(x)
    - Horses are faster than frogs
        - $\forall x,y$ horses(x) ∧ frogs(y)→speed(x) > speed(y)
    - All domesticated horses have an owner
        - $\forall x$ horses(x)^domesticated(x)→$\exists y$ owner(y,x)
    - The rider of a horse can be different from the owner
        - $\forall x,y$ (horses(x)^domesticated(x)) person(y) → ($\exists z$ rider(z,x) ≠ owner(y,x)) ∨ ($\exists z$ rider(z,x) = owner(y,x))
    - A finger is any digit on a hand other than the thumb
        - $\forall x$ finger(x) ↔ digit(x) ∧ onHand(x) ∧ ¬thumb(x)
    - An isosceles triangle is defined as a polygon with 3 edges connected at 3 vertices, where 2 (but not 3) edges have the same length
        - $\forall x$ isoscelesTriangle(x) ↔ polygon(x) ∧ edges(x,3) ∧ vertices(x,3) ∧ $\exists y,z,w$(edge(y,x) ∧ edge(z,x) ∧ edge(w,x) ∧ connects(y,z,x) ∧ connects(z,w,x) ∧ connects(w,y,x) ∧ hasSameLength(y,z,x) ∧ hasSameLength(z,w,x) ∧ ¬hasSameLength(y,w,x)))

2. Convert to **CNF:** $\forall x$ person(x) ^ [$\exists z$ petOf(x,z) ^ $\forall y$ petOf(x,y)→dog(y)] → doglover(x)
    - *Eliminate the implication:* $\forall x$ person(x) ∧ [$\exists z$ petOf(x,z) ∧ $\forall y$(¬petOf(x,y) ∨ dog(y))] → doglover(x)
    - $\forall x$ person(x) ∧ [$\exists z$ petOf(x,z) ∧ $\forall y$(¬petOf(x,y) ∨ dog(y))] → doglover(x)
    - $\forall x$ person(x) ∧ [$\exists z$ petOf(x,z) ∧ $\forall y$ ¬petOf(x,y) ∨ $\forall y$dog(y)] → doglover(x)
    - $\forall x$ person(x) ∧ [petOf(x,z_0) ∧ $\forall y$ ¬petOf(x,y) ∨ $\forall y$dog(y)] → doglover(x)
    - $\forall x$ person(x) ∧ $\exists z_0$[petOf(x,z_0) ∧ $\forall y$ ¬petOf(x,y) ∨ $\forall y$dog(y)] → doglover(x)
    - *Eliminate the implication:* $\forall x$ person(x) ∧ $\exists z_0$[¬(petOf(x,z_0) ∧ $\forall y$ ¬petOf(x,y) ∨ $\forall y$dog(y))] ∨ doglover(x)
    - *Apply DeMorgan's Law:* $\forall x$ person(x) ∧ $\exists z_0$[¬petOf(x,z_0) ∨ $\exists y$petOf(x,y) ∧ $\forall y$¬dog(y)] ∨ doglover(x)
    - $\forall x$ person(x) ∧ $\exists z_0 \exists y$[¬petOf(x,z_0) ∨ (petOf(x,y) ∧ $\forall y$¬dog(y))] ∨ doglover(x)
    - **$\forall$x,z_0,y[person(x) ∧ (¬petOf(x, z_0) ∨ (petOf(x,y) ∧ $\forall$y¬dog(y)))] ∨ doglover(x)**

3. Determine if unifiable:
    - 'owes(owner(X), citibank, cost(X))' and 'owes(owner(ferrari), Z, cost(Y))' are *unifiable*, the unifier is u={X/ferrari, Z/citibank, Y/ferrari}, and the unified expression is 'owes(owner(ferrari), citibank, cost(ferrari))' for both.
    - 'gives(bill, jerri, book21)' and 'gives(X, brother(X), Z)' are **not** *unifiable*, as the second predicate's second argument is a variable, while the first's is a constant. This would work if we *knew* that Jerri was Bill's brother, but since this is not explicitly stated, we cannot be sure.

- ‘opened(X, result(open(X), s0))” and ‘opened(toolbox, Z)’ are *unifiable*, the unifier is u={X/toolbox, Z/result(open(toolbox), s0)}, and the unified expression is ‘opened(toolbox, result(open(toolbox), s0))’ for both.

4. Given the situation:

   - Translate to FOL:
     Marcus is a Pompeian.
       - isPompeian(Marcus)

     All Pompeians are Romans.
       - ∀x (isPompeian(x) → isRoman(x))

     Caesar is a ruler.
       - isRuler(Caesar)

     All Romans are either loyal to Caesar or hate Caesar (but not both).
       - ∀x isRoman(x) → (isLoyalTo(x, Caesar) ⊕ hates(x, Caesar))

     Everyone is loyal to someone.
       - ∀x person(x) → isLoyalTo(x, y)

     People only try to assassinate rulers they are not loyal to.
       - ∀x,y person(x)^triesToAssassinate(x,y)^isRuler(y) → ¬isLoyalTo(x,y)

     Marcus tries to assassinate Caesar.
       - triesToAssassinate(Marcus, Caesar)

   - Prove that Marcus hates Caesar using Natural Deduction. Label all derived sentences with the ROI and which prior sentences and unifier were used.
     - isPompeian(Marcus) → isRoman(Marcus) ; *Marcus is Roman* ***[MP]***
     - triesToAssassinate(Marcus, Caesar) ; *Marcus tries to assassinate Caesar*
     - person(Marcus)^triesToAssassinate(Marcus, Caesar)^isRuler(Caesar) → ¬isLoyalTo(Marcus,Caesar) ; *People only try to assassinate rulers they are not loyal to – Caesar is a ruler, and Marcus tries to assassinate him, so Marcus is not loyal to Caesar* ***[MP]***
     - isRoman(Marcus) → (isLoyalTo(Marcus, Caesar) ⊕ hates(Marcus, Caesar)) ; *Marcus exclusively is either loyal to Caesar or hates Caesar, and we know that he is not loyal to him.* ***[MP]***
     - hates(Marcus, Caesar) ; *therefore, Marcus hates Caesar*

   - Convert all the sentences into CNF
     Marcus is a Pompeian.
       - CNF: isPompeian(Marcus)

     All Pompeians are Romans.
       - ∀x (isPompeian(x) → isRoman(x))
         - CNF: ¬isPompeian(x) v isRoman(x)

     Caesar is a ruler.
       - isRuler(Caesar)

- CNF: isRuler(Caesar)

All Romans are either loyal to Caesar or hate Caesar (but not both).

- ∀x isRoman(x) → (isLoyalTo(x, Caesar) ⊕ hates(x, Caesar))
  - CNF: ¬isRoman(x) ⋁ isLoyalTo(x,Caesar), ¬isRoman(x) ⋁ hates(x,Caesar)

Everyone is loyal to someone.

- ∀x person(x) → isLoyalTo(x, y)
  - CNF: ¬person(x) ⋁ isLoyalTo(x, y)

People only try to assassinate rulers they are not loyal to.

- ∀x,y person(x)^triesToAssassinate(x,y)^isRuler(y) → ¬isLoyalTo(x,y)
  - CNF: ¬person(x) ⋁ ¬triesToAssassinate(x,y) ⋁ ¬isRuler(y) ⋁ ¬isLoyalTo(x,y)

Marcus tries to assassinate Caesar.

- triesToAssassinate(Marcus, Caesar)
  - CNF: triesToAssassinate(Marcus,Caesar)

- Prove that Marcus hates Caesar using Resolution Refutation
  - ¬isPompeian(Marcus) *; Negation of clause: (1)*
  - isRoman(Marcus) ; *Resolution of: (1, 9)*
  - isLoyalTo(Marcus,Caesar) ; *Resolution of: (2, 10)*
  - ¬isRoman(Marcus) ; *Negation of clause: (4)*
  - hates(Marcus,Caesar) ; *Resolution of: (12, 11)*

5. Write a KB in First-Order Logic with rules/axioms for…
   - a. Map-coloring –
     - state(WA), state(NT), state(SA), state(Q), state(T), state(Q), state(NSW), state(V)
     - neighbors(WA, NT), neighbors(WA, SA), neighbors(NT, Q), neighbors(NT, SA), neighbors(SA, Q), neighbors(SA, NSW), neighbors(SA, V), neighbors(NSW, Q), neighbors(NSW, V)
     - color(R), color(G), color(B)
     - ∀s state(s) → ∃c color(c)^hasColor(s,c)
     - ∀s,c,d state(s)^hasColor(s,c)^hasColor(s,d) → c=d
     - ∀s,t,c state(s)^state(t)^neigh(s,t)^hasColor(s,c) → ¬hasColor(t,c)
   - b. Sammy's Sport Shop – include implications of facts like obs(1,W) or label(2,B), as well as constraints about the boxes and colors. Use predicate 'cont(x,q)' to represent that box x contains tennis balls of color q (where q could be W, Y, or B).
     - color(Y), color(W), color(B)
     - box(1), box(2), box(3)
     - ∀x,C obs(x,C) → cont(x,C) v cont(x, B)
     - ∀x,C label(x,C) → cont(x,C) v ¬cont(x,C)

- $\forall x,C$ cont(x,C) → cont(x,C)
- $\forall x1,x2,x3,C1,C2,C3$ cont(x1,C1)^cont(x2, C2) → cont(x3,C3)
- $\forall x,C1,C2$ contains(X, C1)^contains(X, C2), C1 ≠ C2.

c. Wumpus World - (hint start by defining a helper concept 'adjacent(x,y,p,q)' which defines when a room at coordinates (x,y) is adjacent to another room at (p,q). Don't forget rules for 'stench', 'breezy', and 'safe'.

- agent(x, y)
- $\forall x,y,p,q$ (adjacent(x, y, p,q) ↔ (|x - p| = 1 ∧ y = q) ∨ (x = p ∧ |y - q| = 1))
- $\forall x,y$ (stench(x, y) → ($\exists p,q$ (adjacent(x, y, p, q) ∧ Wumpus(p, q))))
- $\forall x,y$ (breezy(x, y) → ($\exists p, q$ (adjacent(x, y, p, q) ∧ Pit(p, q))))
- $\forall x,y$ (safe(x, y) → ¬Pit(x, y) ∧ ¬Wumpus(x, y))
- $\forall x,y$ (safe(x, y) ← ¬Pit(x, y) ∧ ¬Wumpus(x, y))
- $\forall x,y$ (safe(x, y) ← $\forall p, q$ (adjacent(x,y,p,q) → safe(p, q)))
- $\forall x,y$ (¬safe(x, y) ← Pit(x, y) ∨ Wumpus(x, y))
- stench(x, y) ↔ $\exists p,q$ (adjacent(x,y,p,q) ∧ Wumpus(p, q))
- breezy(x, y) ↔ $\exists p,q$ (adjacent(x,y,p,q) ∧ Pit(p, q))

d. 4-Queens – assume row(1)…row(4) and col(1)…col(4) are facts; write rules that describe configurations of 4 queens such that none can attack each other, using 'queen(r,c)' to represent that there is a queen in row r and col c.

- row(1), row(2), row(3), row(4)
- col(1), col(2), col(3), col(4)
- $\forall x,y$ queen(x,y) → row(x)^col(y)
- $\forall r1,r2,c$ queen(r1,c) ^ queen(r2, c) → r1 ≠ r2
- $\forall r,c1,c2$ queen(r,c1) ^ queen(r, c2) → c1 ≠ c2
- $\forall r1, r2, c1, c2$ (queen(r1, c1) ^ queen(r2, c2) → (r1 ≠ r2) ^ (c1 ≠ c2) ^ (|r1 - r2| ≠ |c1 - c2|))