1.

- ∀x bowlingBall(x) → sportEquipment(x)
- ∀x ∀y horse(x) ^ frog(y) → greaterThan(speed(x), speed(y))
- ∀x horse(x) ^ domesticated(x) → ∃y owner(y, x)
- ∃x horse(x) → ∃z ∃y rider(z, x) ^ owner(y, x) ^ notEqual(z, y)
- ∀x digit(x) ^ ∃h hand(h) ^ on(x, h) ^ -thumb(x) → finger(x)
- ∀x isoTri(x) ↔ polygon(x) ^ threeEdges(x) ^ connectedAtThreeVertices(x) ^ ∃e1 ∃e2 ∃e3 equal(length(e1), length(e2)) ^ notEqual(length(e1), length(e3))

2.

∀x person(x) ^ [∃z petOf(x,z) ^ [∀y petOf(x,y)→dog(y)]] → doglover(x)

∀x person(x) ^ [∃z (petOf(x,z) ^ [∀y (petOf(x,y)→dog(y))]] → doglover(x)

∀x -(person(x) ^ [∃z (petOf(x,z) ^ [∀y (petOf(x,y)→dog(y)))]]) v doglover(x)

∀x -person(x) v -[∃z (petOf(x,z) ^ [∀y (petOf(x,y)→dog(y)))]] v doglover(x)

∀x -person(x) v [∀z -(petOf(x,z) ^ [∀y (petOf(x,y)→dog(y)))]] v doglover(x)

∀x -person(x) v [∀z -petOf(x,z) v -[∀y (petOf(x,y)→dog(y))]] v doglover(x)

∀x -person(x) v [∀z -petOf(x,z) v [∃y -(petOf(x,y)→dog(y))]] v doglover(x)

∀x -person(x) v [∀z -petOf(x,z) v [∃y -(-petOf(x,y) v dog(y))]] v doglover(x)

∀x -person(x) v [∀z -petOf(x,z) v [∃y petOf(x,y) ^ -dog(y)]] v doglover(x)

∀x ∀z ∃y -person(x) v -petOf(x,z) v [petOf(x,y) ^ -dog(y)] v doglover(x)

y = f(x,z)

∀x ∀z -person(x) v -petOf(x,z) v [petOf(x,f(x,z)) ^ -dog(f(x,z))] v doglover(x)

Drop quantifiers since CNF assumes universal by default

Final:

(-person(x) v -petOf(x,z) v petOf(x,f(x,z)) v doglover(x)) ^ (-person(x) v -petOf(x,z) v -dog(f(x,z)) v doglover(x))

3.

- Unifiable
  - {X/ferrari, Z/citibank, Y/ferrari}
  - owes(owner(ferrari),citibank,cost(ferrari)
- Not Unifiable
  - X → bill, would need brother(bill) → jerry
  - Since brother(bill) does not literally equal ‘jerry’ we can’t substitute. One is constant and the other is function term.
- Unifiable
  - {X/toolbox, Z/result(open(toolbox),s0)}
  - opened(toolbox,result(open(toolbox),s0))

4.

1. pompeian(marcus)
2. ∀x pompeian(x) → roman(x)
3. ruler(ceaser)
4. ∀x roman(x) → (loyal(x, ceasar) ⊕ hate(x, ceasar))
5. ∀x ∃y loyal(x,y)
6. ∀x ∀y tryToAssassinate(x,y) → (ruler(y) ^ -loyal(x,y))
7. tryToAssassinate(marcus, ceasar)

8. roman(marcus) // MP 2,1
9. ruler(ceasar) ^ -loyal(marcus, ceasar) // MP 6,7
10. -loyal(marcus, ceasar) // AE 9
11. roman(marcus) → (loyal(marcus, ceasar) ⊕ hate(marcus, ceasar)) // UI 4
12. loyal(marcus, ceasar) ⊕ hate(marcus, ceasar) // MP 11,8
13. hate(marcus, ceasar) // XOR 12, 10

splitting ∀x roman(x) → (loyal(x, ceasar) ⊕ hate(x, ceasar)) into 2 clauses for the ⊕
splitting ∀x ∀y tryToAssassinate(x,y) → (ruler(y) ^ -loyal(x,y)) into 2 clauses for the ^

1. pompeian(marcus)
2. -pompeian(x) v roman(x)
3. ruler(ceasar)
4. -roman(x) v loyal(x, ceasar) v hate(x, ceasar)
5. -roman(x) v -loyal(x, ceasar) v -hate(x, ceasar)
6. loyal(x,f(x)) // Skolemized
7. -tryToAssassinate(x,y) v ruler(y)
8. -tryToAssassinate(x,y) v -loyal(x,y)
9. tryToAssassinate(marcus, ceasar)

q. hate(marcus, ceasar)
-q. -hate(marcus, ceasar)

10. roman(marcus) // Resolution 2,1
11. -loyal(marcus, ceasar) // Resolution 8,9
12. -roman(marcus) v loyal(marcus, ceasar) v hate(marcus, ceasar) // UI 4
13. loyal(marcus, ceasar) v hate(marcus, ceasar) // Resolution 12,10
14. hate(marcus, ceasar) // Resolution 13,11
15. ø // Resolution 14, -q, empty clause which means we proved KB ⊨ q

5.

Map Coloring

// Each state has at least one color
$\forall s\ state(s) \rightarrow \exists c\ color(c) \wedge hasColor(s,c)$
// Each state has at most one color
$\forall s\ \forall c1\ \forall c2\ state(s) \wedge color(c1) \wedge color(c2) \wedge hasColor(s,c1) \wedge hasColor(s,c2) \rightarrow equal(c1,c2)$
// Adjacent states can't share a color
$\forall s\ \forall t\ \forall c\ state(s) \wedge state(t) \wedge neigh(s,t) \wedge hasColor(s,c) \rightarrow -hasColor(t,c)$

Sammy's Sport Shop

// Exactly 3 boxes with different content types
$\exists x\ \exists y\ \exists z\ notEqual(x,y) \wedge notEqual(y,z) \wedge notEqual(x,z) \wedge cont(x,W) \wedge cont(y,Y) \wedge cont(z,B)$
// Each box has one content type
$\forall x\ \forall c1\ \forall c2\ cont(x,c1) \wedge cont(x,c2) \rightarrow equal(c1,c2)$
// Label is incorrect
$\forall x\ \forall c\ label(x,c) \rightarrow -cont(x,c)$
// Observation means at either that color or both
$\forall x\ obs(x,W) \rightarrow (cont(x,W) \vee cont(x,B))$
$\forall x\ obs(x,Y) \rightarrow (cont(x,Y) \vee cont(x,B))$
// B box has both colors
$\forall x\ cont(x,B) \rightarrow \exists b1\ \exists b2\ (obs(x,b1) \wedge obs(x,b2) \wedge notEqual(b1,b2))$

Wumpus World

Helper:
$\forall x\ \forall y\ \forall p\ \forall q\ adjacent(x,y,p,q) \leftrightarrow ((|x-p| = 1 \wedge y = q) \vee (|y-q| = 1 \wedge x = p))$
// Wumpus check
$\forall x\ \forall y\ stench(x,y) \leftrightarrow \exists p\ \exists q\ (adjacent(x,y,p,q) \wedge wumpus(p,q))$
// Pit check
$\forall x\ \forall y\ breeze(x,y) \leftrightarrow \exists p\ \exists q\ (adjacent(x,y,p,q) \wedge pit(p,q))$
// Safe check
$\forall x\ \forall y\ safe(x,y) \leftrightarrow -pit(x,y) \wedge -wumpus(x,y) \wedge -\exists p, -\exists q\ adjacent(x,y,p,q) \wedge (pit(p,q) \vee wumpus(p,q))$

4 Queens

// Each row has at least one queen

$\forall r \; row(r) \rightarrow \exists c \; col(c) \wedge queen(r,c)$

// Exactly one queen per row

$\forall r \; \forall c1 \; \forall c2 \; queen(r,c1) \wedge queen(r,c2) \rightarrow equal(c1,c2)$

// Exactly one queen per column

$\forall c \; \forall r1 \; \forall r2 \; queen(r1,c) \wedge queen(r2,c) \rightarrow equal(r1,r2)$

// Queens can’t share a diagonal

$\forall r1 \; \forall r2 \; \forall c1 \; \forall c2 \; (queen(r1,c1) \wedge queen(r2,c2) \wedge notEqual(r1,r2)) \rightarrow notEqual(|r1-r2|,|c1-c2|)$