Problem 1

English Sentence	First Order Logic
bowling balls are sporting equipment	∀x ball(x, bowling) ^ equipment(x, sport)
horses are faster than frogs	∀x,y horse(x)^frog(y) → faster(x, y)
all domesticated horses have an owner	∀x,y horse(x)^domesticated(x) → owner(y, x) (owner(y,x) means x is owned by y)
the rider of a horse can be different than the owner	∃x,y horse(x)^rider(x,y)^¬owner(y) (rider(x,y) means y is a rider of x)
a finger is any digit on a hand other than the thumb	∃x digit(x)^on(x, hand)^¬thumb(x) → finger(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	∃t,x,y,z polygon(t)^edgeOf(t, x)^edgeOf(t, y)^edgeOf(t, z)^vertice(x,y)^vertice(x, z)^vertice(y,z)^Equal(x, y)^¬Equal(x, z)^¬Equal(y, z)

Problem 2

Convert the following first-order logic sentence into CNF:

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

Eliminate implications

- ∀xperson(x)∧[∃zpetOf(x,z)∧∀y(¬petOf(x,y)∨dog(y))]→doglover(x)

Move negations inward

- ∀xperson(x)∧[∃zpetOf(x,z)∧∀y(¬petOf(x,y)∨dog(y))]∨doglover(x)

Standardize

- ∀xperson(x)∧[∃zpetOf(x,z)∧∀w(petOf(x,w)∧dog(w))]∨doglover(x)

Skolemize + Drop universal quantifiers

- person(x)∧[petOf(x,F(x))∧petOf(x,G(x))∧dog(w)]∨doglover(x)

Distribute ^ over v

- person(x)∧petOf(x,F(x))∧(petOf(x,G(x))∧dog(g(x)))∨doglover(x)

Problem 3

Predicates	Unifier and Unified Expression
owes(owner(X),citibank,cost(X)) owes(owner(ferrari),Z,cost(Y))	u={X/ferrari, Y/X, Z/citibank} owes(owner(ferrari),citibank,cost(ferrari)
gives(bill, jerry, book21) gives(X,brother(X),Z)	No unifier
opened(X,result(open(X),s0))) opened(toolbox,Z)	u={X/toolbox,Z/result(open(toolbox),s0)} opened(toolbox,result(open(toolbox),s0))

<u>Problem 4</u>

a) Translate the sentences to First-Order Logic

English Sentence	**FOL**
Marcus is a Pompeian.	1. pompeian(marcus)
All Pompeians are Romans.	2. ∀x pompeian(x) → roman(x)
Ceasar is a ruler.	3. ruler(caesar)
All Romans are either loyal to Caesar or hate Caesar (but not both).	4. ∀x roman(x) →((loyal(x, caesar) ⇔ ¬hate(x, caesar))^(¬loyal(x, caesar) ⇔ hate(x, caesar))
Everyone is loyal to someone.	5. ∀x(∃y loyal(x, y))
People only try to assassinate rulers they are not loyal to.	6. ∀x,y assassinate(x, y) → ¬loyal(x, y)
Marcus tries to assassinate Caesar.	7. assassinate(marcus, caesar)

b) Prove that Marcus hates Caesar using Natural Deduction. Label all derived sentences with the ROI and which prior sentences and unifier were used.

1. Start with assassinate(marcus, caesar) (7)
2. ¬loyal(marcus, caesar) [MP 6, 7], u = {x/marcus, y/caesar}
3. roman(marcus) [UnivInst 1] [MP 1, 2],
4. ¬loyal(marcus, caesar) ⇔ hate(marcus, caesar) [MP prev, 4] [AE] u = {x/marcus}
5. **hate(marcus, caesar)** [MP, prev, 2]

c) Convert all the sentences into CNF

FOL	CNF
pompeian(marcus)	1. pompeian(marcus)
∀x pompeian(x) → roman(x)	2. ¬pompeian(x) v roman(x) [IE]
ruler(caesar)	3. ruler(caesar)
∀x roman(x) →((loyal(x, caesar) ⇔ ¬hate(x, caesar))^(¬loyal(x, caesar) ⇔ hate(x, caesar))	4. ¬roman(x) v (¬(loyal(x, caesar) v ¬hate(x, caesar))^(loyal(x, caesar) v hate(x, caesar)) a. ¬(loyal(x, caesar) v ¬hate(x, caesar) b. loyal(x, caesar) v hate(x, caesar)
∀x(∃y loyal(x, y))	5. loyal(x, F(X))
∀x,y assassinate(x, y) → ¬loyal(x, y)	6. ¬assassinate(x, F(x)) v ¬loyal(x, G(x)) [IE]
assassinate(marcus, caesar)	7. assassinate(marcus, caesar)

d) Prove that Marcus hates Caesar using Resolution Refutation.

1. Start with assassinate(marcus, caesar) (7)
2. ¬assassinate(marcus, caesar) v ¬loyal(marcus, caesar), u = {x/marcus, y/caesar} (6)
3. ¬loyal(marcus, caesar) (Resolve 2 and 1)
4. Take loyal(marcus, caesar) v hate(marcus, caesar) u={x/marcus} (4a)
5. **hate(marcus, caesar)** (Resolve (4a and 3))

5. Write a KB in First-Order Logic with rules/axioms for…

a. Map-coloring – every state must be exactly 1 color, and adjacent states must be different colors. Assume possible colors are states are defined using unary predicate like color(red) or state(WA). To say a state has a color, use a binary predicate, e.g. ‘color(WA,red)’.

1. $\forall x \exists c(color(x,c) \rightarrow (\forall d(color(x,d) \rightarrow d=c)))$
2. $\forall x \forall y(adjacent(x,y) \rightarrow \forall c(color(x,c) \rightarrow \neg color(y,c)))$

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).

1. obs(1,W), obs(2, W), obs(3, Y), label(1, W), label(2, Y), label(3, B)
2. $\forall x,q$ label(x, q) $\rightarrow$ ¬cont(x, q)
3. $\forall x,c$ obs(x, c) $\rightarrow$ cont(x, c) v cont(x, B) (where c is Y or W, cannot observe B)

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’.

1. $\forall x \forall y \forall p \forall q$ adjacent(x,y,p,q) $\Leftrightarrow$ $((|x-p|=1 \wedge y=q) v (x=p \wedge |y-q|=1))$
    a. if the difference between x/y coords is 1 and either the x/y coords are equal then the rooms are adjacent
2. $\forall x \forall y$ stench(x,y) $\rightarrow$ $\exists p \exists q$ adjacent(x,y,p,q) $\wedge$ wumpus(p,q)
3. $\forall x \forall y$ breezy(x,y) $\rightarrow$ $\exists p \exists q$ adjacent(x,y,p,q) $\wedge$ pit(p,q)
4. $\forall x \forall y$ safe(x,y) $\Leftrightarrow$ ¬(pit(x,y) $\vee$ wumpus(x,y))

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.

1. $\forall r$ (row(r) $\rightarrow$ $\exists c$ queen(r,c) $\wedge$ $\forall d$ queen(r,d) $\rightarrow$ d=c (one queen in a column)
2. $\forall c$(col(c) $\rightarrow$ $\exists r$queen(r,c) $\wedge$ $\forall e$ queen(e,c) $\rightarrow$ e=r (one queen in a row)
3. $\forall r \forall c$ queen(r,c) $\rightarrow \neg \exists i \exists j$ queen(i,j) $\wedge$ |i−r|=|j−c|) (one queen in a diagonal)