Question 1

FIRST ORDER LOGIC

line1:- is_in_Pawnee(leslieknope) ^ is_in_Pawnee(ronswanson) ^ is_in_Pawnee(andydwyer) ^ is_in_Pawnee(burtmacklin) ^ is_in_Pawnee(annperkins)

line2:- nurse(annperkins) ^ fbi_agent(burtmacklin)

line3:- administrator(ronswanson) ^ administrator(leslieknope)

line4:- forall(X):nurse(X) → wears(X,uniform)

line5:- forall(X):fbi_agent(X) → wears(X,uniform)

line6:- ~administrator(andydwyer) ^ ~wears(andydwyer,uniform)

line7:- exists(X): is_in_Pawnee(X) ^ fbi_agent(X) ^ ~pays(X,rent)

line8:- forall(X): is_in_Pawnee(X) ^ (~nurse(X) v ~administrator(X) v ~fbi_agent(X)) → lives_in(X,pit)

line9:- forall(X): is_in_Pawnee(X) ^ lives_in(X,pit) → ~pays(X,rent)

CLAUSE FORM

line1:- is_in_Pawnee(leslieknope) ^ is_in_Pawnee(ronswanson) ^ is_in_Pawnee(andydwyer) ^ is_in_Pawnee(burtmacklin) ^ is_in_Pawnee(annperkins)
(split cunjuncts into separate clauses)

is_in_Pawnee(leslieknope)

is_in_Pawnee(ronswanson)

is_in_Pawnee(andydwyer)

is_in_Pawnee(burtmacklin)

is_in_Pawnee(annperkins)

line2:- nurse(Annperkins) ^ fbi_Agent(Burtmacklin)

(split conjuncts into separate clauses)

nurse(annperkins)

fbi_agent(aurtmacklin)

line3:- administrator(Ronswanson) ^ administrator(Leslieknope)
(split conjunts into separate clauses)
administrator(ronswanson)

administrator(leslieknope)

line4:- forall(X):nurse(X) → wears(X,uniform)
(toss implications:- forall(x): ~nurse(X) v wears(X,uniform))
(eliminate universal quantifiers:- ~nurse(X) v wears(X,uniform)

~nurse(A) v wears_uniforms(A)

line5:- forall(X):fbi_agent(X) → wears(X,uniform)
(toss implications:- forall(x): ~fbi_agent(X) v wears(X,uniform))
(eliminate universal quantifiers:- ~fbi_agent(X) v wears(X,uniform)

~fbi_agent(B) v wears_uniforms(B)

line6:- ~administrator(andydwyer) ^ ~wears(andydwyer,uniform)
(split conjunts into separate clauses)

~administrator(andydwyer)

~wears_uniform(andydwyer)

line7:- exists(X): is_in_Pawnee(X) ^ fbi_agent(X) ^ ~pays(X,rent)
eliminate existential quantifier:- is_in_Pawnee(bob) ^ fbi_agent(bob) ^ ~pays(bob,rent))

is_in_Pawnee(bob) ^ fbi_agent(bob) ^ ~pays(bob,rent)

line8:- forall(X): is_in_Pawnee(X) ^ (~nurse(X) v ~administrator(X) v ~fbi_agent(X)) → lives_in(X,pit)
toss implication step 1:- forall(X): ~(is_in_Pawnee(X) ^ (~nurse(X) v ~administrator(X) v ~fbi_agent(X)) v lives_in(X,pit)
toss implication step 2 :- forall(X): ~is_in_Pawnee(X) v (nurse(X) ^ administrator(X) ^ fbi_agent(X)) v lives_in(X,pit)

eliminate universal quantifier:- ~is_in_Pawnee(X) v (nurse(X) ^ administrator(X) ^ fbi_agent(X)) v lives_in(X,pit)

~is_in_Pawnee(C) v (nurse(C) ^ administrator(C) ^ fbi_agent(C)) v lives_in(C,pit)

line9:- forall(X): is_in_Pawnee(X) ^ lives_in(X,pit) → ~pays(X,rent)
toss implication step 1:- forall(X): ~(is_in_Pawnee(X) ^ lives_in(X,pit)) v pays(X,rent)
toss implication step 2:- forall(X): ~is_in-Pawnee(X) v ~live_in(X,pit) v pays(x,rent)
(eliminate universal quantifier:- ~is_in-Pawnee(D) v ~live_in(D,pit) v pays(D,rent)

~is_in-Pawnee(D) v ~live_in(D,pit) v pays(D,rent)

Prove that Andydwyer does not pay rent (using resolution refutation)

Negated H:- pays(andydwyer,rent)

Step 1:- Resolve is_in_Pawnee(andydwyer) and ~is_in_Pawnee(C) v (nurse(C) ^ administrator(C) ^ fbi_agent(C)) v lives_in(C,pit) {andydwyer/C}
Which gives you:- nurse(andydwyer) ^ administrator(andydwyer)^fbi_agent(andydwyer) v lives_in(andydwyer, pits)

step 2:- Resolve ~administrator(andydwyer) and nurse(andydwyer) ^ administrator(andydwyer)^fbi_agent(andydwyer) v lives_in(andydwyer, pits)
Which gives you:- lives_in(andydwyer,pits)

Step 3:- Resolve lives_in(andydwyer,pits) and ~is_in_Pawnee(D) v ~live_in(D,pit) v pays(D,rent) {andydwyer/D)
Which gives you:- ~is_in_Pawnee(andydwyer) v pays(andydwyer, rent)

Step4;- Resolve ~is_in_Pawnee(andydwyer) v pays(andydwyer, rent) and is_in_Pawnee(andydwyer)
Which gives you:- pays(andydwyer, rent)

Can resolve no further, thus the negated H is false, which makes H true. Andy does not pay rent.

Question 2

First Order Logic

line 1:- person(mister_fantastic) ^ person(the_invisible_woman) ^ person(magneto) ^ person(wolverine)
line 2:- fantastic_four(mister_fantastic) ^ fantastic_four(the_invisible_woman)
line 3:- has_special_powers(wolverine) ^ ~likes(wolverine,magneto)
line 4:- likes(magneto,mister_Fantastic)
line 5:- forall(X): x_men(X) → ~like(magneto,X)
line 6:- forall(X): ~x_men(X) ^ mutant(X)→ like(X,magneto)
line 7:- forall(X):fantastic_four(X) v has_special_powers(X) → mutant(X)
line 8:- exist(X): ~lucky(X)
line 9:- forall(X): mutant(X) ^ ~x_men(X) → likes(the_invisible_woman,X)
line 10:- forall(X,Y): person(X) ^ person(Y) ^ fight(X,Y) ^ wins(X,Y) → ~wins(Y,X)
line 11:- forall(X,Y):- x_men(X) ^ fantastic_four(Y) ^ fight(X,Y) ^ wins(X,Y) → ~lucky(Y)
line 12:- fight(wolverine,mister_fantastic) ^ wins(wolverine,mister_fantastic)

Clause Form

line 1:- person(mister_Fantastic) ^ person(the_Invisible_Woman) ^ person(magneto) ^ person(wolverine)
(split cunjuncts into separate clauses)
person(mister_fantastic)
person(the_invisible_woman)
person(magneto)
person(wolverine)

line 2:- fantastic_four(mister_fantastic) ^ fantastic_four(the_invisible_woman)

(split cunjuncts into separate clauses)
fantastic_four(mister_fantastic)
fantastic_four(the_invisible_woman)

line 3:- has_special_powers(wolverine) ^ ~likes(wolverine,magneto)
(split cunjuncts into separate clauses)
has_special_powers(wolverine)
~like(wolverine,magneto)

line 4:- likes(magneto,mister_Fantastic)

line 5:- forall(X): x_men(X) → ~like(magneto,X)
toss implications:- forall(x): ~x_men(X) v ~like(magneto,X)
eliminate universal quantifier:- ~x_men(X) v ~like(magneto,X)
~x_men(A) v ~like(magneto,A)

line 6:- forall(X): ~x_men(X) ^ mutant(X)→ like(X,magneto)
toss implications step1:- forall(X): ~(~x_men(X) ^ mutant(X)) v like(X,magneto)
toss implications step2:- forall(X): x_men(X) v ~mutant(X) v like(X,magneto)
eliminate universal quantifier:- x_men(X) v ~mutant(X) v like(X,magneto)
x_men(B) v ~mutant(B) v like(B,magneto)

line 7:- forall(X):fantastic_four(X) v has_special_powers(X) → mutant(X)
toss implications step1:- forall(X): ~(fantastic_four(X) v has_special_powers(X)) v mutant(X)
toss implications step2:- forall(X): ~fantastic_four(X) ^ ~has_special_powers(X) v mutant(X)
eliminate universal quantifier:- ~fantastic_four(X) ^ ~has_special_powers(X) v mutant(X)
~fantastic_four(C) ^ ~has_special_powers(C) v mutant(C)

line 8:- exist(X): ~lucky(X)
eliminate existential quantifier
~lucky(somebody)

line 9:- forall(X): mutant(X) ^ ~x_men(X) → likes(the_invisible_woman,X)
toss implications step1:- forall(X): ~(mutant(X) ^ ~x_men(X)) v likes(the_invisible_woman,X)
toss implications step2:- forall(X): ~mutant(X) v x_men(x) v likes(ths_invisible_woman,X)
eliminate universal quantifier:-~mutant(X) v x_men(x) v likes(the_invisible_woman,X)
~mutant(D) v x_men(D) v likes(the_invisible_woman,D)

line 10:- forall(X,Y): person(X) ^ person(Y) ^ fight(X,Y) ^ wins(X,Y) → ~wins(Y,X)
toss implications step1:- forall(X,Y): ~(person(X) ^ person(Y) ^ fight(X,Y) ^ wins(X,Y)) v ~wins(Y,X)
toss implications step2:- forall(X,Y): ~person(X) v ~person(Y) v ~fight(X,Y) v ~wins(X,Y) v ~wins(Y,X)
eliminate universal quantifier:- ~person(X) v ~person(Y) v ~fight(X,Y) v ~wins(X,Y) v ~wins(Y,X)
~person(E) v ~person(F) v ~fight(E,F) v ~wins(E,F) v ~wins(F,E)

line 11:- forall(X,Y):- x_men(X) ^ fantastic_four(Y) ^ fight(X,Y) ^ wins(X,Y) → ~lucky(Y)
toss implications step1:- forall(X,Y):- ~(x_men(X) ^ fantastic_four(Y) ^ fight(X,Y) ^ wins(X,Y)) v ~lucky(Y)
toss implications step2:- forall(X,Y):- ~x_men(X) v ~fantastic_four(Y) v ~fight(X,Y) v ~wins(X,Y) v ~lucky(Y)
eliminate universal quantifier:- ~x_men(X) v ~fantastic_four(Y) v ~fight(X,Y) v ~wins(X,Y) v ~lucky(Y)
~x_men(J) v ~fantastic_four(K) v ~fight(J,K) v ~wins(J,K) v ~lucky(K)

line 12:- fight(wolverine,mister_fantastic) ^ wins(wolverine,mister_fantastic)
(split cunjuncts into separate clauses)
fight(wolverine,mister_fantastic)
wins(wolverine,mister_fantastic)

I) Prove that Mister Fantastic is not lucky

lucky(mister_fantastic) – ~H

Step 1:- Resolve lucky(mister_fantastic) and ~x_men(J) v ~fantastic_four(K) v ~fight(J,K) v ~wins(J,K) v ~lucky(K)
which gives:- ~x_men(J) v ~fantastic_four(mister_fantastic) v ~fight(J,mister_fantastic) v ~wins(J,mister_fantastic) {mister_fantastic/K}

Step 2:- Resolve fantastic_four(mister_fantastic) and ~x_men(J) v ~fantastic_four(mister_fantastic) v ~fight(J,mister_fantastic) v ~wins(J,mister_fantastic) {mister_fantastic/K}

which gives:- ~x_men(J) v ~fight(J,mister_fantastic) v ~wins(J,mister_fantastic)

Step 3:- Resolve fight(wolverine,mister_fantastic) and ~x_men(J) v ~fight(J,mister_fantastic) v ~wins(J,mister_fantastic) {wolverine/J}
which gives:- ~x_men(wolverine) v ~wins(wolverine,mister_fantastic)

Step 4:- Resolve x_men(wolverine) and ~x_men(wolverine) v ~wins(wolverine,mister_fantastic)
which gives you:- ~wins(wolverine,mister_fantastic)

Step 5:- Resolve wins(wolverine,mister_fantastic) and ~wins(wolverine,mister_fantastic)
which gives you:- nil which in turn means the negated H is false, thus making H true. Mister Fantastic is not lucky.

II) Prove that the invisible woman likes mister fantastic

~likes(the_invisible_woman,mister_fantastic) - ~H

Step1:- Resolve fantastic_four(mister_fantastic) and ~fantastic_four(C) ^ ~has_special_powers(C) v mutant(C) {mister_fantastic/C}
which give you:- mutant(mister_fantastic)

Step2:- Resolve likes(magneto,mister_Fantastic) and ~x_men(A) v ~like(magneto,A) {mister_fantastic/A}
which gives you:- ~x_man(mister_fantastic)

Step3: Resolve mutant(mister_fantastic) and ~mutant(D) v x_men(D) v likes(the_invisible_woman,D) {mister_fantastic/D}
which gives you:- x_men(mister_fantastic) v likes(the_invisible_woman, mister_fantastic)

Step4: Resolve ~x_men(mister_fantastic) and x_men(mister_fantastic) v likes(the_invisible_woman, mister_fantastic)
which gives you:- likes(the_invisible_woman, mister_fantastic)
Step 5:- Resolve likes(the_invisible_woman, mister_fantastic) and ~likes(the_invisible_woman,mister_fantastic)
Which gives you:- nil, which in turn means the negated H is false, thus making H true. The Invisible Woman likes Mister Fantastic

III)Prove that someone is not an X-man

FOL : exist(X): ~x_men(X)
Clause form: ~x_men(someone)
Negated H: x_men(X)

Step 1:- Resolve x_man(someone) and ~x_men(A) v ~like(magneto,A) {someone/A}
Which gives you:- ~like(magneto,someone)

Step2:- Unable to resolve due to lack of information about whether this someone is a mutant or not.

IV)Prove that someone is not a mutant

FOL: exist(X): ~mutant(X)

Clause form: ~mutant(someone)
Negated H: mutant(someone)

Step 1:- Resolve mutant(someone) and x_men(B) v ~mutant(B) v like(B,magneto) {someone/B}
Which gives you:- x_men(someone) v like(someone,magneto)

Step 2:- Resolve x_men(someone) v like(someone,magneto) and ~x_men(A) v ~like(magneto,A) {someone/A}
Which gives you:- like(someone,magneto) v ~like(magneto,someone)

Question 3

English

1. Pit, Scrap, fluffy, fido and tweety are pets
2. Pit, scrap and fido are dogs
3. fluffy is a cat and tweety is a bird
4. All dogs hate cats
5. All cats have claws
6. Some cats don't hate birds
7. Some cats eat birds
8. If a cat is hungry it will eat a bird
9. fluffy is hungry
10. someone ate tweety

First order logic

1. pet(pit) ^ pet(scrap) ^ pet(fluffy) ^ pet(fido) ^ pet(tweety)
2. dog(pit) ^ dog(scrap) ^ dog(fido)
3. cat(fluffy) ^ bird(tweety)
4. forall(X): dog(X) → hates(X,cat)
5.forall(X): cat(X) → hasClaws(X)
6. exists(X): cat(X) → hates(X,birds)
7. exists(X): cat(X) → eats(X,bird)
8. forall(X): cat(X) ^ hungry(X) → eats(X,bird)
9. hungry(fluffy)

10.exist(X): eats(X,tweety)

Clause Form

1. pet(pit) ^ pet(scrap) ^ pet(fluffy) ^ pet(fido) ^ pet(tweety)
(split cunjuncts into separate clauses)
pet(pit)
pet(scrap)
pet(fluffy)
pet(fido)
pet(tweety)

2. dog(pit) ^ dog(scrap) ^ dog(fido)
(split cunjuncts into separate clauses)
dog(pit)
dog(scrap)
dog(fido)

3. cat(fluffy) ^ bird(tweety)
(split cunjuncts into separate clauses)
cat(fluffy)
bird(tweety)

4. forall(X): dog(X) → hates(X,cat)
toss implications step1:- forall(X):- ~dog(X) v hate(X,cats)
eliminate universal quantifier:- ~dog(X) v hate(X,cats)
~dog(A) v hates(A,cats)

5.forall(X): cat(X) → hasClaws(X)
toss implications step1:- forall(X):- ~cat(X) v hasClaws(X)
eliminate universal quantifier:- ~cat(X) v hasClaws(X)
~cat(B) v hasClaws(B)

6. exists(X): cat(X) → hates(X,birds)
toss implications step1:- exist(X):- ~cat(X) v hates(X,bird)
eliminate existential quantifier:-
~cat(fred) v hates(fred,bird)

7. exists(X): cat(X) → eats(X,bird)
toss implications step1:- exist(X):- ~cat(X) v eats(X,bird)
eliminate existential quantifier:-

~cat(toby) v eats(toby,bird)

8. forall(X): cat(X) ^ hungry(X) → eats(X,bird)
toss implications step1:- forall(X):- ~cat(X) v ~hungry(X) v eats(X,bird)
eliminate universal quantifier:-
~cat(C) v ~hungry(C) v eats(C,bird)

9. hungry(fluffy)

10.exist(X): eats(X,tweety)
eliminate existential quantifier:-
eats(somebody, tweety)

Prove that Fluffy ate tweet

Negated H : ~eat(fluffy,tweety)

Step 1:- Resolve ~eat(fluffy,tweety) and ~cat(C) v ~hungry(C) v eats(C,bird) {fluffy/C}
Which gives you:- ~cat(fluffy) v ~hungry(fluffy)

Step2 :- Resolve cat(fluffy) and ~cat(fluffy) v ~hungry(fluffy)
Which gives you:- ~hungry(fluffy)

Step 3:- Resolve hungry(fluffy) and ~hungry(fluffy)
Which gives you:- nil, which means the negative H is false, which means fluffy did eat tweety.