CS161 HW5

1.

a. neither, three worlds satisfy the statement (satisfiable) but it is not always true in all worlds (not valid)

Smoke	Fire	Smoke => Fire	-Smoke	-Fire	-Smoke => -Fire	(Smoke ⇒ Fire) ⇒ (-Smoke ⇒ -Fire)
T	T	T	F	F	T	T
T	F	F	F	T	T	T
F	T	T	T	F	F	F
F	F	T	T	T	T	T

b. neither, statement is satisfiable but not valid since there is one world where it is not true

Smoke	Fire	Smoke => Fire	Heat	Smoke ∨ Heat	(Smoke ∨ Heat) ⇒ Fire	(Smoke ⇒ Fire) ⇒ ((Smoke ∨ Heat) ⇒ Fire)
T	T	T	T	T	T	T
T	F	F	T	T	F	T
F	T	T	T	T	T	T
F	F	T	T	T	F	F
T	T	T	F	T	T	T
T	F	F	F	F	T	T
F	T	T	F	T	T	T
F	F	T	F	F	T	T

c. valid since it is true for all worlds in truth table

Smoke	Heat	Fire	Smoke ∧ Heat	((Smoke ∧ Heat) ⇒ Fire)	((Smoke ⇒ Fire) ∨ (Heat ⇒ Fire)	((Smoke ∧ Heat) ⇒ Fire) ⇔ ((Smoke ⇒ Fire) ∨ (Heat ⇒ Fire))
T	T	T	T	T	T	T
T	T	F	T	F	F	T
F	T	T	F	T	T	T
F	T	F	F	T	T	T
T	F	T	F	T	T	T
T	F	F	F	T	T	T
F	F	T	F	T	T	T
F	F	F	F	T	T	T

2.

a. Knowledge Base: mythical(unicorn) => immortal(unicorn),
-mythical(unicorn) => -immortal(unicorn) ∧ mammal(unicorn),
immortal(unicorn) ∨ mammal(unicorn) => horned(unicorn),
horned(unicorn) => magical(unicorn),

b. Convert to CNF: (-mythical(unicorn) ∨ immortal(unicorn)) ∧

(mythical(unicorn) ∨ (-immortal(unicorn) ∧ mammal(unicorn))) ∧
(-(immortal(unicorn) ∨ mammal(unicorn)) ∨ horned(unicorn)) ∧
(-horned(unicorn) ∨ magical(unicorn)
=
(-mythical(unicorn) ∨ immortal(unicorn)) ∧
(mythical(unicorn) ∨ (-immortal(unicorn)) ∧
(mythical(unicorn) ∨ mammal(unicorn)) ∧
((-immortal(unicorn) ∧ -mammal(unicorn)) ∨ horned(unicorn)) ∧
(-horned(unicorn) ∨ magical(unicorn))
=
(-mythical(unicorn) ∨ immortal(unicorn)) ∧
(mythical(unicorn) ∨ (-immortal(unicorn)) ∧
(mythical(unicorn) ∨ mammal(unicorn)) ∧
(-immortal(unicorn) ∨ horned(unicorn)) ∧
(-mammal(unicorn) ∨ horned(unicorn)) ∧
(-horned(unicorn) ∨ magical(unicorn))

c. No, it is not possible to prove that the unicorn is mythical.
(-mythical(unicorn) ∨ immortal(unicorn)) ∧
(mythical(unicorn) ∨ (-immortal(unicorn)) ∧
(mythical(unicorn) ∨ mammal(unicorn)) ∧
(-immortal(unicorn) ∨ horned(unicorn)) ∧
(-mammal(unicorn) ∨ horned(unicorn)) ∧
(-horned(unicorn) ∨ magical(unicorn)) ∧
(-mythical(unicorn))
=>

(-immortal(unicorn)) ∧	because (mythical(unicorn) ∨ (-immortal(unicorn))
mammal(unicorn))	because (mythical(unicorn) ∨ mammal(unicorn))
=>	
-mythical(unicorn) ∧	because (-mythical(unicorn) ∨ immortal(unicorn))
horned(unicorn))	because (-mammal(unicorn) ∨ horned(unicorn))
=>	
magical(unicorn)	done, cannot continue with resolution

Yes, it is possible to prove magical by refutation.
(-mythical(unicorn) ∨ immortal(unicorn)) ∧
(mythical(unicorn) ∨ (-immortal(unicorn)) ∧
(mythical(unicorn) ∨ mammal(unicorn)) ∧
(-immortal(unicorn) ∨ horned(unicorn)) ∧
(-mammal(unicorn) ∨ horned(unicorn)) ∧
(-horned(unicorn) ∨ magical(unicorn)) ∧
(-magical(unicorn))

=>

-horned(unicorn)

=>

-mammal(unicorn) ∧ -immortal(unicorn)

=>

mythical(unicorn) ∧	because (mythical(unicorn) ∨ mammal(unicorn))
-mythical(unicorn)	because (-mythical(unicorn) ∨ immortal(unicorn))
=>	
magical(unicorn)	by contradiction

Yes, it is possible to prove horned.

(-mythical(unicorn) ∨ immortal(unicorn)) ∧
(mythical(unicorn) ∨ (-immortal(unicorn)) ∧
(mythical(unicorn) ∨ mammal(unicorn)) ∧
(-immortal(unicorn) ∨ horned(unicorn)) ∧
(-mammal(unicorn) ∨ horned(unicorn)) ∧
(-horned(unicorn) ∨ magical(unicorn)) ∧
(-horned(unicorn))

=> -mammal(unicorn) ∧ -immortal(unicorn)

mythical(unicorn) ∧	because (mythical(unicorn) ∨ mammal(unicorn))
-mythical(unicorn	because (-mythical(unicorn) ∨ immortal(unicorn))
=>	
horned(unicorn)	by contradiction

3.

a. {x/A, y/B, z/B} unifies P(A, B, B), P(x, y, z)

b. unifier does not exist for Q(y, G(A, B)), Q(G(x, x), y)

c. {x/John, y/John} unifies Older(Father(y), y), Older(Father(x), John) because y = John, Father(x) = Father(y), x = John

d. unifier does not exist for knows(Father(y),y), Knows(x,x) because x = y, x = Father(y), y = Father(y)

4.

a.

A x, Food(x) => Likes(John, x)
Food(Apple)
Food(Chicken)
A x, E y, Eats(y, x) & -Killedby(y, x) => Food(x)
A y, x, Killedby(y, x) => -Alive(y)
Eats(Bill, Peanuts) & Alive(Bill)
A x, Eats(Bill, x) => Eats(Sue, x)

b.

(- Food(x) | Likes(John, x)) &

Food(Apple) &
Food(Chicken) &
-Eats(y, x) | Killedby(y, x) | Food(x) &
- Killedby(y, x) | -Alive(y) &
Eats(Bill, Peanuts) & Alive(Bill) &
-Eats(Bill, x) | Eats(Sue, x)

because A x, E y, Person(y) & Eats(y, x) & -Killedby(y, x) => Food(x) can be changed to:
-(Person(y) & Eats(y, x) & -Killedby(y, x)) | Food(x)
= -Person(y) | -Eats(y, x) | Killedby(y, x) | Food(x)

c. Proof that John likes peanuts given knowledge base:
(- Food(x) | Likes(John, x)) &
Food(Apple) &
Food(Chicken) &
-Eats(y, x) | Killedby(y, x) | Food(x) &
- Killedby(y, x) | -Alive(y) &
Eats(Bill, Peanuts) & Alive(Bill) &
-Eats(Bill, x) | Eats(Sue, x) &

Unification on (-Eats(y, x) | Killedby(y, x) | Food(x)) with Eats(Bill, Peanuts) by {x/Peanuts, y/Bill}:
    -Eats(Bill, Peanuts) | Killedby(Bill, Peanuts) | Food(Peanuts)
Resolution with Eats(Bill, Peanuts):
    Killedby(Bill, Peanuts) | Food(Peanuts)
Unification with - Killedby(y, x) | -Alive(y) by {x/Peanuts, y/Bill}:
    -Killedby(Bill, Peanuts) | -Alive(Bill)
Resolution with Alive(Bill):
    -Killedby(Bill, Peanuts)
Resolution with Killedby(Bill, Peanuts) | Food(Peanuts):
    Food(Peanuts)
Unification with - Food(x) | Likes(John, x) by {x/Peanuts}:
    - Food(Peanuts) | Likes(John, Peanuts)
Resolution with Food(Peanuts):
    Likes(John, Peanuts)

d. What food does Sue eat:
Unification on Eats(Bill, Peanuts) with -Eats(Bill, x) | Eats(Sue, x) by {x/Peanuts}
    -Eats(Bill, Peanuts) | Eats(Sue, Peanuts)
Resolution with Eats(Bill, Peanuts):
    Eats(Sue, Peanuts)

e. What food does Sue eat, with added knowledge: A x, -Eat(x) => Die(x), Die(x) => -Alive(x), Alive(Bill)

New knowledge base:
(- Food(x) | Likes(John, x)) &
Food(Apple) &
Food(Chicken) &
-Eats(y, x) | Killedby(y, x) | Food(x) &
- Killedby(y, x) | -Alive(y) &
(Eats(x, y) |Die(x)) &
(-Die(x) | -Alive(x)) &
Alive(Bill) &
-Eats(Bill, x) | Eats(Sue, x)

Unification on Alive(Bill) and -Die(x) | -Alive(x) with {x/Bill}:

    -Die(Bill) | -Alive(Bill)

Resolution with Alive(Bill):

    -Die(Bill)

Unification on Eats(x, y) | Die(x) and -Die(Bill) with {x/Bill, y/k} where K is a skolem constant:

    Eats(Bill, K) | Die(Bill)

Resolution with –Die(Bill):

    Eats(Bill, K)

Unification on -Eats(Bill, x) | Eats(Sue, x) with {x/K}

    -Eats(Bill, K) | Eats(Sue, K)

Resolution with Eats(Bill, K):

    Eats(Sue, K)   ie, Sue eats something