Erikson Sodergren
Intro to IS: Homework 3

1.
   a. Paris and Marseilles are both in France
      i. In(Paris ∧ Marseilles,France): **syntactically invalid**
      ii. In(Paris,France) ∧ In(Marseilles,France): **correct**
      iii. In(Paris,France) ∨ In(Marseilles,France): **valid but incorrect**
   b. There is a country that borders both Iraq and Pakistan.
      i. ∃ c Country(c) ∧ Border (c,Iraq) ∧ Border (c,Pakistan): **correct**
      ii. ∃ c Country(c) ⇒ [Border (c,Iraq) ∧ Border (c,Pakistan)]: **valid but incorrect**
      iii. [∃ c Country(c)] ⇒ [Border (c,Iraq) ∧ Border (c,Pakistan)]: **syntactically invalid**
      iv. ∃ c Border (Country(c),Iraq ∧ Pakistan): **syntactically invalid**
   c. All countries that border Ecuador are in South America
      i. ∀ c Country(c) ∧ Border (c,Ecuador) ⇒ In(c, SouthAmerica): **correct**
      ii. ∀ c Country(c) ⇒ [Border (c,Ecuador) ⇒ In(c, SouthAmerica)]: **correct**
      iii. ∀ c [Country(c) ⇒ Border (c,Ecuador)] ⇒ In(c, SouthAmerica): **valid but incorrect**
      iv. ∀ c Country(c) ∧ Border (c,Ecuador) ∧ In(c, SouthAmerica): **valid but incorrect**
2.
   a. Occupation(Emily, surgeon) V Occupation(Emily, lawyer)
   b. ∃x x != actor ^ Occupation(Joe, actor) ^ Occupation(Joe, x)
   c. ∀x Occupation(x, surgeon) -> Occupation(x, doctor)
   d. !∃x Occupation(x, lawyer) ^ Customer(Joe, x)
   e. ∃x Boss(x, Emily) ^ Occupation(x, lawyer)
   f. ∃x ∀y Occupation(x, lawyer) ^ Customer(y, x) -> Occupation(y, doctor)
   g. ∀x OCcupation(x, surgeon) -> ∃y Occupation(y, lawyer) ^ Customer(x, y)
3.
   a. KB:
      ∀x cat(x) -> animal(x)
      ∀x,y [animal(x) ^ afraid(x, y)] -> run(x, y) V hide(x, y)
      ∀x,y cat(x) ^ dog(y) -> afraid(x, y)
      ∀x,y cat(x) ^ car(y) -> afraid(x, y)
      cat(Louie)
      dog(Jake)
      ∀x,y animal(x) ^ hide(x, y) -> !seen(x)
      seen(Louie)
   b. KB in CNF
      1: !cat(x) V animal(x)
      2: !animal(x) V !afraid(x, y) V run(x, y) V hide(x, y)

3: !cat(x) V !dog(y) V afraid(x, y)
4: !cat(x) V !car(y) V afraid(x, y)
5: cat(Louie)
6: dog(Jake)
7: !animal(x) V !hide(x, y) V !seen(x)
8: seen(Louie)

c. Proof:
    i. 1: !run(Louie, Jake)
    ii. 2: 1 + KB2 = !animal(Louie) V !afraid(Louie, Jake) V hide(Louie, Jake)
    iii. 3: KB1 + KB5 = animal(Louie)
    iv. 4: 2+3 = !afraid(Louie, Jake) V hide(Louie, Jake)
    v. 5: KB8+KB7 = !animal(Louie) V !hide(Louie, y)
    vi. 6: 5+3 = !hide(Louie, y)
    vii. 7: 6+4 = !afraid(Louie, Jake)
    viii. 8: KB3 + KB5 = !dog(y) V afraid(Louie, y)
    ix. 9: 8 + KB6 = afraid(Louie, Jake)
    x. 10: 7 + 9 = []

4.
a. ?- setof(X,grandchild(X,elizabeth),List).
   List = [beatrice, eugenie, harry, james, louise, peter, william, zara].
b. ?- setof(X,brotherInLaw(X,diana),List).
   List = [andrew, edward].
c. ?- setof(X,greatgrandparent(X,zara),List).
   List = [george, mum].
d. ?- setof(X,ancestor(X,eugenie),List).
   List = [andrew, elizabeth, george, mum, philip, sarah].

5.
a. Action(Move(loc1,loc2,r),
        Precond: at(loc1) ^ in(loc1, r) ^ in(loc2, r)
        Effect: at(loc2) ^ !at(loc1))
   Action(Push(b,loc1,loc2,r),
        Precond: at(loc1) ^ box(b) ^ in(loc1, r) ^ in(loc2, r) ^ boxAt(b, loc1)
        Effect: at(loc2) ^ !at(loc1) ^ boxAt(b, loc2) ^ !boxAt(b, loc1))
   Action(TurnOn(s),
        Precond: at(s) ^ !on(s)
        Effect: on(s))
   Action(TurnOff(s),
        Precond: at(s) ^ on(s)
        Effect: !on(s))
b. at(Loc0) ^ in(Loc0, Room2) ^ boxAt(Box1, Loc1) ^ boxAt(Box2, Loc2) ^ boxAt(Box3, Loc3) ^ in(Loc1, Room1) ^ in(Loc2, Room1) ^ in(Loc3, Room3) ^ box(Box1) ^ box(Box2) ^ box(Box3) ^ in(Door1, Room1) ^ in(Door1, Hall) ^

in(Door2, Room2) ^ in(Door2, Hall) ^ in(Door3, Room3) ^ in(Door3, Hall) ^ on(Light1) ^ on(Light2) ^ on(Light3) ^ in(Light1, Room1) ^ in(Light2, Room2) ^ in(Light3, Room3)

c. Move(Loc0, Door2, Room2), Move(Door2, Door1, Hall), Move(Door1, Loc1, Room1), Push(Box1, Loc1, Door1, Room1), Push(Box1, Door1, Door3, Hall)
d. Maximum branching factor in this case is if all 3 boxes were in Door1, with the robot. This allows the robot to move to other doors, move to Light1, move to Loc1/Loc2, or push any of the 3 boxes to one of these locations for a total of 5locations * 4options(alone or with any box) = 20.
    i. This is not purely dependent upon number of boxes and rooms, but also upon locations defined in a room, but the maximum would be with all boxes beginning in the same room and therefore needing locations defined in that room to begin at. This gives said room m+1 non-door locations to move to, k-1 doors to move to, and m boxes to possibly push, for a total of ((m+1)+(k-1))*(m+1) = (m+k)*(m+1) = (m^2+mk+m+k) branches