1. *Does a KB with P(e) and P(b) entail for any x, P(x)*
No. Consider any model with three domain elements, where e and b refer to the first two elements and the relation referred to by P holds only for those two elements.

2. *A FOL statement s.t. every world in which it is true has exactly one object.*
∀x,y x=y, under the usual (canonical) semantics of equality.

3. *Write FOL statements.*
   a) ∃b Barber(b) ∧ (∀m (Man(m) ∧ ¬Shaves(m,m)) → Shaves(m,b))
   b) ∀x (Person(x) ∧ ¬Born(x, Florin) ∧ (∃y Parent(y, x) ∧ Citizen(y, Florin, Birth)) → Citizen(x, Florin, Descent)).
   c) ∀x Politician(x) => (∃y ∀t Person(y) ∧ Fools(x,y,t) ) ∧ (∃t ∀y Person(y) => Fools(x,y,t) ) ∧ ¬(∀t ∀y Person(y) => Fools(x,y,t))

4. *Derive the most general unifier if it exists:* 
   i) (if assuming already standardized apart [shouldn’t do this]) {(Q(y, G(A,B)), Q(G(x,x), y)) → { } (y, G(x,x)), (G(A,B), y)} → { } (y, G(x,x)), (G(A,B), G(x,x)) → { } (y, G(x,x)), (A, x), (B, x)) → { } (y, G(x,x)), (A, x), (B, A)) 
   Fails, since B cannot be unified with A.
   If not standardized apart: {(Q(y, G(A,B)), Q(G(x,x), z)) → { } (y, G(x,x)), (G(A,B), z)}: The MGU is Q(G(x,x), G(A,B)).
   ii) {(Knows(Father(y), y), Knows(x,x))} → { } (Father(y), x), (y, x)) → { } (Father(y), y), (y, Father(y))) → fails due to the Occurs check. Standardizing apart makes no difference here.

5. *If the occurs check is omitted from the unification algorithm* 
∀x∃yP(x,y), from Skolemization, we have ∀xP(x, f(x))
   
   KB = ∀xP(x, f(x)), C = ∃rP(r, r)

   $\theta = \{ x/r, f(x)/r \}$
   
   $P(r, r) \land \neg P(r, r)$

   Example:
∀x, Child(x, Mother(x)) ⊢ ∀x, Child(x, x)

6. **Riddle “Brothers and sister”**
   Suppose the KB has Son(Me, MF), Father(MS, Me) and Father(Me, MF) where Me denotes myself, MF my father and MS denotes my son. The query is: ∃x,z,r Father(x,z) ∧ Son(z,r) ∧ Father(Me,r) (here x is “that man”, so “there is a man x with father z, and there is an r who is my father, and z is r’s son). Negating the query yields ¬Father(x,z) ∨ ¬Son(z,r) ∨ ¬Father(Me,r). Resolving with Father(MS, Me), Son(Me, MF) and Father(Me, MF) yields the empty clause, with x bound to MS. So x is my son. Note that here, by having just a single fact Father(Me, MF), we are implicitly saying “no brothers or sisters.” (We also saying no brothers or sisters for MS, which is stronger than what is given but irrelevant for our purposes.) This could be done in a more elaborate way, by explicitly including those formulas (i.e. “My father has a son,” “I have no brothers”, “I have no sisters” etc.)

7. **Resolution on Ancestor KB**
   C1= Ancestor(Mother(x), x)
   C2=Ancestor(x, y)∧Ancestor(y, z) ⇒ Ancestor(x, z)
   C=¬Ancestor(John, John)

   C1∧C2∧¬C

   Ancestor(Mother(x), x)∧(¬Ancestor(x, y)∨¬Ancestor(y, z)∨Ancestor(x, z))∧Ancestor(John, John)

   θ = {x/z}

   Ancestor(Mother(z), z)∧(¬Ancestor(z, y)∨¬Ancestor(y, z)∨Ancestor(z, z))∧Ancestor(John, John)

   θ = {z/John}

   Ancestor(Mother(John), John)∧

   (¬Ancestor(John, y)∨¬Ancestor(y, John)∨Ancestor(John, John))∧Ancestor(John, John)

   No further unification can be performed. Thus, we couldn’t conclude C.

8. **Monkey and Banana problem:**
   a)
Init (At(Monkey,A) AND At(Bananas,B) AND At(Box,C) AND Height(Bananas,Tall) AND Height(Monkey,Short) AND Height(Box,Short))

b)
Action (Push(box,from,to)),
PRECOND: At(Monkey,from)
Effect: At(box,to) AND At(monkey,to) AND ~At(box,from) AND ~At(box,to)

Action (Go(from,to))
PRECOND: At(Monkey,from)
Effect: At(Monkey,to) AND ~At(Monkey, from)

Action ClimbUp(y)
PRECOND: At(box,y) AND At(monkey,y) AND Level(monkey,low)
Effect: Level(Monkey,high) AND ~Level(monkey,low)

Action ClimbDown(y)
PRECOND: At(box,y) AND At(monkey,y) AND Level(monkey,high)
Effect: Level(Monkey,low) AND ~Level(monkey,high)

Action Grab(object,y,x)
PRECOND: At(object,y) AND At(monkey,y) AND Level(object,x) AND Level(monkey, x)
Effect: Hold(monkey,object)

Action Drop(object,y,l)
PRECOND: At(monkey,y) AND Hold(monkey,object) AND Level(monkey,l)
Effect: \(~\text{Hold(monkey,object)} \land \text{At(object,y)} \land \text{Level(object,l)}\)

9. **Dropping negative effects from action schemas**
   Negative effects result in “undoing” previously solved literals, and so complicate the problem. If there are no negative effects, a simple linear approach: pick the first goal literal, solve it, then the second, solve it, etc... is guaranteed to find a solution since a solved literal will never be deleted. (Note that goals can’t contain negative literals and action preconditions are positive as well.) It should be fairly obvious that the cost of this solution will be a lower bound on the cost of the true solution that includes negative effects that potentially undo solved subgoal literals. Hence, removing negative effects results in a relaxed problem.

10. **Bidirectional search in State space planning and plan space planning**
    1. Yes bidirectional search could be used in state space planning. One just starts from the initial state and search forward, while also starting from goal state and search backward. If two searches meet each other, then we have found the plan. However, it may not be efficient when the state space is large. Therefore bidirectional search is possible but not necessarily a good idea.

    2. No, plan space planning can’t be done in a bidirectional manner because each state is a POP, so a backward search would start from the goal state, which is a complete plan. And if we know this we wouldn’t need to plan! However, in usual POP, we add actions starting with the preconditions of the END action. Actually we could also have search operators that add actions starting from the START. This is the combining ideas from state space planning and POP, and performing “bidirectional search” within a state for POP. This is doable.