Question 1

20 points.

For of the following pairs of sentences, determine if they are logically equivalent (i.e., if each sentence of the pair implies the other sentence in the pair). Do not assume anything except the laws of propositional and first order logic.

1a.
```
for-every x, exists y: color(x) = y
not (exists x, for-every y: not (color(x) = y) )
```

```
logically equivalent
```

1b.
```
for-every x, exists y: f(x, y)
exists y, for-every x,: f(x, y)
```

```
not logically equivalent
```

1c.
```
for-every x, exists y: color(x) = y
Not (for-every x, exists y: not (color(x) = y) )
```

```
not logically equivalent
```

1d.
```
for-every x, exists y: son(x) = y
for-every x, exists y: father(y) = x
```

```
not logically equivalent
```

Question 2

20 points.

Determine what is the most general unifier for each of the following pairs of expressions.
The following conventions hold:
F and G are relations.
x, y, z are variables.
John and Mary are grounded symbols.

2a. `F(x, y), F(y, x)`

{x/y}

2b. `F(x, y), F(John, z)`

{x/John, y/z}

2c. `F(x, y, z), F(z, G(Mary), John)`

{x/Z, y/G(Mary), z/John}

2d. `F(x, g(y)), G(John, Mary)`

No unification is possible here.

Question 3

30 points.

3a. Consider the following set of actions:

```
Action(PutSockOnFoot(a, f):
          Precond: Sock(a), Foot(f), FreeSock(a)
          Effect: not (FreeSock(a)), SockOn(f))


Action(PutShoeOnFoot(b, f):
          Precond: Shoe(b), Foot(f), SockOn(f), FreeShoe(b)
          Effect:  not (FreeShoe(b)), ShoeOn(f))
```

Now, consider this initial state, and this goal:

```
InitState: Sock(sock1) and Sock(sock2) and
           FreeSock(sock1)and FreeSock(sock2) and
            Shoe(left_shoe) and Shoe(right_shoe) and
           FreeShoe(left_shoe)and FreeShoe(right_shoe)
           and Foot(left_foot) and Foot(right_foot)

Goal: ShoeOn(left_foot) and ShoeOn(right_foot)
```

Make two different totally-ordered plans to achieve the goal, given the initial state.

First plan:

PutSockOnFoot(sock1, left_foot)
PutSockOnFoot(sock2, right_foot)
PutShoeOnFoot(left_shoe, left_foot)
PutShoeOnFoot(right_shoe, right_foot)

Second plan:

PutSockOnFoot(sock1, left_foot)
PutShoeOnFoot(left_shoe, left_foot)
PutSockOnFoot(sock2, right_foot)
PutShoeOnFoot(right_shoe, right_foot)

3b. Consider the following set of actions (just a little different from **3a**).

```
Action(PutSockOnFoot(a, f):
          Precond: Sock(a), Foot(f)
          Effect: SockOn(f))


Action(PutShoeOnFoot(b, f):
          Precond: Shoe(b), Foot(f), SockOn(f)
          Effect:  ShoeOn(f))
```

Now, consider this initial state, and this goal:

```
InitState: Sock(sock1) and Shoe(left_shoe)
           and Foot(left_foot) and Foot(right_foot)

Goal: ShoeOn(left_foot) and ShoeOn(right_foot)
```

Make a totally-ordered plan to achieve the goal, given the initial state.

PutSockOnFoot(sock1, left_foot)
PutShoeOnFoot(left_shoe, left_foot)
PutSockOnFoot(sock1, right_foot)
PutShoeOnFoot(left_shoe, right_foot)

3c. Consider the following set of actions (exactly the same as **3b**).

```
Action(PutSockOnFoot(a, f):
          Precond: Sock(a), Foot(f)
          Effect: SockOn(f))


Action(PutShoeOnFoot(b, f):
          Precond: Shoe(b), Foot(f), SockOn(f)
          Effect:  ShoeOn(f))
```

Now, consider this initial state, and this goal (also exactly the same as **3b**).

```
InitState: Sock(sock1) and Shoe(left_shoe)
           and Foot(left_foot) and Foot(right_foot)

Goal: ShoeOn(left_foot) and ShoeOn(right_foot)
```

Make a partially-ordered plan to achieve the goal, given the initial state. No order should be imposed between any two actions unless necessary (in other words, the plan should be minimally ordered).

$$\begin{tikzcd}
 & \text{START} \arrow[dl] \arrow[dr] & \\
\text{PutSockOnFoot(sock1, left\_foot)} \arrow[d] & & \text{PutSockOnFoot(sock1, right\_foot)} \arrow[d] \\
\text{PutShoeOnFoot(left\_shoe, left\_foot)} \arrow[dr] & & \text{PutShoeOnFoot(left\_shoe, right\_foot)} \arrow[dl] \\
 & \text{FINISH} &
\end{tikzcd}$$

3d. Consider the following set of actions (slightly different than **3c**).

```
Action(PutSockOnFoot(a, f):
          Precond: Sock(a), Foot(f)
          Effect: SockOn(f) or SockOnFloor(a))


Action(PutShoeOnFoot(b, f):
          Precond: Shoe(b), Foot(f), SockOn(f)
          Effect:  ShoeOn(f))
```

Now, consider this initial state, and this goal (exactly the same as **3b** and **3c**).

```
InitState: Sock(sock1) and Shoe(left_shoe)
           and Foot(left_foot) and Foot(right_foot)

Goal: ShoeOn(left_foot) and ShoeOn(right_foot)
```

Is there a finite conditional plan that always achieves the goal given the initial state? If yes, describe the plan. If not, why not?

There is no finite conditional plan that always achieves the goal. There is no guarantee that, after any finite number of repeating the PutSockOnFoot(sock1, left_foot) action, we will achieve SockOn(left_foot), which is a precondition for putting a shoe on the left foot.

3e. Consider the following set of actions (exactly the same as **3d**).

```
Action(PutSockOnFoot(a, f):
          Precond: Sock(a), Foot(f)
          Effect: SockOn(f) or SockOnFloor(a) )


Action(PutShoeOnFoot(b, f):
          Precond: Shoe(b), Foot(f), SockOn(f)
          Effect:  ShoeOn(f) )
```

Now, consider this initial state, and this goal (exactly the same as **3b**, **3c**, and **3d**).

```
InitState: Sock(sock1) and Shoe(left_shoe)
           and Foot(left_foot) and Foot(right_foot)

Goal: ShoeOn(left_foot) and ShoeOn(right_foot)
```

Provide an execution-monitoring plan to achieve the goal given the initial state.

```
while(not(SockOn(left_foot)))
  PutSockOnFoot(sock1, left_foot)
end

while(not(SockOn(right_foot)))
  PutSockOnFoot(sock1, right_foot)
end

PutShoeOnFoot(left_shoe, left_foot)
PutShoeOnFoot(right_shoe, right_foot)
```

3f. Consider the following set of actions (exactly the same as in **3a**):

```
Action(PutSockOnFoot(a, f):
          Precond: Sock(a), Foot(f), FreeSock(a)
          Effect: not (FreeSock(a)), SockOn(f))


Action(PutShoeOnFoot(b, f):
          Precond: Shoe(b), Foot(f), SockOn(f), FreeShoe(b)
          Effect:  not (FreeShoe(b)), ShoeOn(f))
```

Now, consider this initial state, and this goal (also exactly the same as in **3a**)

```
InitState: Sock(sock1) and Sock(sock2) and
           FreeSock(sock1)and FreeSock(sock2) and
            Shoe(left_shoe) and Shoe(right_shoe) and
           FreeShoe(left_shoe)and FreeShoe(right_shoe)
           and Foot(left_foot) and Foot(right_foot)

Goal: ShoeOn(left_foot) and ShoeOn(right_foot)
```

Give an example of a plan (sequence of actions) that does not make much intuitive sense (a person would not normally do such a thing, or would find it pretty awkward to do such a thing). What modifications to the actions and init state are needed to fix that?

PutSockOnFoot(sock1, left_foot)
PutShoeOnFoot(right_shoe, left_foot)
PutSockOnFoot(sock2, right_foot)
PutShoeOnFoot(left_shoe, right_foot)

To prevent putting on the wrong shoe on each foot, we can create a predicate Compatible, that is true for shoes and feet that match. Then, we would add precondition Compatible(b, f) to PutShoeOnFoot, and we would add to init state the statements

Compatible(left_shoe, left_foot)
Compatible(right_shoe, right_foot)

Question 4 - 30 points

4a. (5 points) Consider the following knowledge base:

A AND B
C OR D
(A => (C OR D)) AND (NOT (A => C))

How many rows are there in the truth table for this knowledge base? How did you determine this number?

There are four symbols, A, B, C, D, so there are 2^4=16 rows in the truth table.

4b. (5 points) Which sentences, if any, do you obtain by applying the resolution inference rule to the following pair of sentences? **Do not do any simplifications to either the input or the output sentences, just blindly apply the resolution rule.**

A OR (NOT B) OR C OR (NOT D)
(NOT B) OR (NOT C) OR D OR H

1. A OR (NOT B) OR (NOT D) OR (NOT B) OR D OR H
2. A OR (NOT B) OR C OR (NOT B) OR (NOT C) OR H

4c. (5 points) Which sentences, if any, do you obtain by applying the resolution inference rule to the following pair of sentences? **Do not do any simplifications to either the input or the output sentences, just blindly apply the resolution rule.**

A OR (NOT B) OR C OR (NOT D)
(NOT B) OR C OR (NOT G) OR H

The resolution rule is not applicable here.

4d. (5 points) Put the following knowledge base in conjunctive normal form:

A => (B OR NOT C)
C OR (A AND (NOT B))

(NOT A) OR B OR (NOT C)
C OR A
C OR (NOT B)

4e. (5 points) John and Mary sign the following binding contract in front of their parents:
1. On Sunday, John will mow the lawn or buy groceries.
2. On Sunday, Mary will mow the lawn or wash the car.

This is an all-inclusive list of what actually happens on Sunday:

1. Mary mows the lawn on Sunday.
2. Mary washes the car on Sunday.
3. Mary buys groceries on Sunday.

How can the above statements be represented using propositional logic? First, define literals and specify what English phrase each literal corresponds to. Second, represent the knowledge base (i.e., what happens on Sunday) using those literals. Third, represent the contract as a single logical statement, using those literals. Four, determine (in any way you like) whether, according to the rules of propositional logic, the contract was violated or not.

We define the following literals:
JM: John mows the lawn on Sunday
JB: John buys groceries on Sunday
MB: Mary buys groceries on Sunday
MM: Mary mows the lawn on Sunday
MW: Mary washes the car on Sunday

This is what happens on Sunday:
MM and MW and MB and (not JM) and (not JB)

Note: (not JM) and (not JB) are included because the question explicitly says that the given list of what happens on Sunday is all-inclusive (so, whatever is not specified there, did not happen).

The contract is represented as:
(JM or JB) and (MM or MW)

The contract was violated, since (JM or JB) was false.

4f. (5 points, harder) Suppose that a knowledge base contains only symbols A, B, and C. When does such a knowledge base entail the statement D (i.e., the statement consisting of a single symbol that does not appear in the knowledge base)? Always, sometimes, or never? If sometimes, then identify precisely the conditions that determine whether this knowledge base entails the statement D.

There is only one case where this knowledge base entails D: the case where the knowledge base is always false. A knowledge base that is always false entails everything.