1a. Implication Introduction

X	Y	Z	¬X ∨ Z ∨ ¬Y	X ∧ Y	(X ∧ Y) → Z
F	F	F	T	F	T
F	F	T	T	F	T
F	T	F	T	F	T
F	T	T	T	F	T
T	F	F	T	F	T
T	F	T	T	F	T
T	T	F	F	T	F
T	T	T	T	T	T

We can see that, in all cases where the Horn Clause (¬X ∨ Z ∨ ¬Y) is true, so too is the implication ((X ∧ Y) → Z). This is because the implication will always be true if X or Y is false, and will only be false if X and Y are true and Z is false, due to the laws of classical logic. The only case where the implication is false here is when X=1, Y=1, and Z=0. Therefore, the implication introduction is a sound rule of inference.

1b.

A	B	C	D	(A ∧ B) → (C ∧ D)	(A ∧ B) → C
F	F	F	F	T	T
F	F	T	F	T	T
F	F	F	T	T	T
F	F	T	T	T	T
F	T	F	F	T	T
F	T	F	T	T	T
F	T	T	F	T	T
F	T	T	T	T	T
T	F	F	F	T	T

T	F	F	T	T	T
T	F	T	F	T	T
T	F	F	F	T	T
T	T	F	F	F	F
T	T	T	F	F	T
T	T	F	T	F	F
T	T	T	T	T	T

As you can see, in all cases where "(A ∧ B) → (C ∧ D)" is true (1), "(A ∧ B) → C" is also true. There is indeed a case where "(A ∧ B) → (C ∧ D)" is false and "(A ∧ B) → C" is true, but every single time the first is true, the second is too.

1c. To prove (A ∧ B → C ∧ D) |= (A ∧ B → C) using Natural Deduction, we follow the steps below:

(1) Assume A ∧ B → C ∧ D, and A ∧ B
(2) Using 1a, we can transform A ∧ B into A ∧ B → C (we can use this since I have proven it is sound)
(3) Apply Modus Ponens to combine A ∧ B → C and A ∧ B, which gives us C. We know A and B are true, so we can deduct that C is as well
(4) Get rid of the assumption in step 2 (A ∧ B), as it has been proven
(5) We have now shown that if A ∧ B → C ∧ D holds, then A ∧ B → C holds. Thus, (A ∧ B → C ∧ D) |= (A ∧ B → C).

1d. Resolution

- ¬(A ∧ B → C)
- ¬(¬(A ∧ B) ∨ C) (implication rule)
- (A ∧ B) ∧ ¬C (De Morgan's Law)
- (A ∧ B → C ∧ D) (Given premise in the problem statement)

Begin Resolution:

- (A ∧ B → C ∧ D) ∨ ¬(A ∧ B → C) (Negation)
- (¬(A ∧ B) ∨ C ∧ D) ∨ (¬(¬(A ∧ B) ∨ C)) (Rewrite (A ∧ B → C) with new disjunctions)
- (¬(A ∧ B) ∨ C ∧ D) ∨ ((A ∧ B) ∧ ¬C) (Double negation introduced and De Morgan's Law)
- ((¬A ∨ ¬B) ∨ C ∧ D) ∨ (A ∧ B ∧ ¬C) (Double negation elimination)

Apply Resolution using complementary literals:

- $(\neg A \lor \neg B) \lor (A \land B)$ (Resolving $\neg C$ with $C \land D$)
- $\neg A \lor \neg B \lor A$ (Resolving A with $\neg C$)
- $\neg A \lor \neg B$ (Using Resolution to eliminate A)

Now we have $\neg A \lor \neg B$, therefore $\neg(A \land B)$ is true. However, this contradicts the given premise $(A \land B \rightarrow C \land D)$ because this premise assumes that $A \land B$ implies $C \land D$.

Since we've reached a contradiction, our initial assumption that $\neg(A \land B \rightarrow C)$ is false, which means that $(A \land B \rightarrow C)$ is valid using Resolution.

2a. Sammy.kb = {

(C1W v C2W v C3W) ∧ (C1Y v C2Y v C3Y) ∧ (C1B v C2B v C3B)
(L1W v L2W v L3W) ∧ (L1Y v L2Y v L3Y) ∧ (L1B v L2B v L3B)

(L1W ∧ L2B) → L3Y
(L1W ∧ L2Y) → L3B
(L1Y ∧ L2W) → L3B
(L1Y ∧ L2B) → L3W
(L1B ∧ L2W) → L3Y
(L1B ∧ L2Y) → L3W
(L1W ∧ L3B) → L2Y
(L1W ∧ L3Y) → L2B
(L1Y ∧ L3W) → L2B
(L1Y ∧ L3B) → L2W
(L1B ∧ L3W) → L2Y
(L1B ∧ L3Y) → L2W
(L2W ∧ L3B) → L1Y
(L2W ∧ L3Y) → L1B
(L2Y ∧ L3W) → L1B
(L2Y ∧ L3B) → L1W
(L2B ∧ L3W) → L1Y
(L2B ∧ L3Y) → L1W

(O1W ∧ O2W) → C3Y
(O1Y ∧ O2Y) → C3W
(O2Y ∧ O3Y) → C1W
(O2W ∧ O3W) → C1Y
(O1Y ∧ O3Y) → C2W

$(O1W \land O3W) \rightarrow C3Y$

$(O1Y \land O2W) \lor (O1W \land O2Y) \rightarrow (O3Y \lor O3W) \land (C3W \lor C3Y \lor C3B)$
$(O1Y \land O3W) \lor (O1W \land O3Y) \rightarrow (O2Y \lor O2W) \land (C2W \lor C2Y \lor C2B)$
$(O2Y \land O3W) \lor (O2W \land O3Y) \rightarrow (O1Y \lor O1W) \land (C1W \lor C1Y \lor C1B)$

$(C1W \land C2Y) \lor (C1Y \land C2W) \rightarrow C3B$
$(C2W \land C3Y) \lor (C2Y \land C3W) \rightarrow C1B$
$(C1W \land C3Y) \lor (C1Y \land C3W) \rightarrow C2B$
$(C1W \land C2B) \lor (C1B \land C2W) \rightarrow C3Y$
$(C2W \land C3B) \lor (C2B \land C3W) \rightarrow C1Y$
$(C1W \land C3B) \lor (C1B \land C2W) \rightarrow C2Y$
$(C1B \land C2Y) \lor (C1Y \land C2B) \rightarrow C3W$
$(C2B \land C3Y) \lor (C2Y \land C3B) \rightarrow C1W$
$(C1B \land C3Y) \lor (C1Y \land C2B) \rightarrow C2W$

$O1Y \leftrightarrow (C1Y \lor C1B)$
$O1W \leftrightarrow (C1W \lor C1B)$
$O2Y \leftrightarrow (C2Y \lor C2B)$
$O2W \leftrightarrow (C2W \lor C2B)$
$O3Y \leftrightarrow (C3Y \lor C3B)$
$O3W \leftrightarrow (C3W \lor C3B)$

}

2b. Prove that C2W is true, given:

- $L1W \land L2Y \land L3B$
- $O1Y \land O2W \land O3Y$

In this situation, though our kb does have rules for labels, we know that the boxes are labeled incorrectly. Therefore, our deduction will be purely based on the observations made when pulling a random ball from a particular box. As we can see, a *yellow* was drawn from box 1, a *white* from box 2, and a *yellow* from box 3 ($O1Y \land O2W \land O3Y$). In the kb that was defined above in 2a, we can see that we have a rule for this: $(O1Y \land O3Y) \rightarrow C2W$. This means that if a yellow ball is observed from box 1 and box 3, box 2 is implied to contain only *white* balls. This is because balls can only contain yellow, white, or a combination of the two balls.

If a yellow ball is drawn from both 1 and 3, one of those will contain all yellow balls, and one will contain a combination of the two; it doesn't matter which is which. This is illustrated with the following rules from our kb:

- O1Y ↔ (C1Y ∨ C1B)
- O3Y ↔ (C3Y ∨ C3B)

As we can see, they can each only contain either all yellow balls, or both. For completeness' sake, an additional rule is referenced for box 2:

- O2W ↔ (C2W ∨ C2B)

We can clearly observe that the middle box must be either white or a combination of both. But, since one between box 1 (O1Y) and box 2 (O3Y) *have to be both already*, the middle must contain only white balls, confirming that C2W is true.

2c. kb to CNF:

{

(C1W v C2W v C3W) ∧ (C1Y v C2Y v C3Y) ∧ (C1B v C2B v C3B)
(L1W v L2W v L3W) ∧ (L1Y v L2Y v L3Y) ∧ (L1B v L2B v L3B)

(¬L1W v ¬L2B v L3Y)
(¬L1W v ¬L2Y v L3B)
(¬L1Y v ¬L2W v L3B)
(¬L1Y v ¬L2B v L3W)
(¬L1B v ¬L2W v L3Y)
(¬L1B v ¬L2Y v L3W)
(¬L1W v ¬L3B v L2Y)
(¬L1W v ¬L3Y v L2B)
(¬L1Y v ¬L3W v L2B)
(¬L1Y v ¬L3B v L2W)
(¬L1B v ¬L3W v L2Y)
(¬L1B v ¬L3Y v L2W)
(¬L2W v ¬L3B v L1Y)
(¬L2W v ¬L3Y v L1B)
(¬L2Y v ¬L3W v L1B)
(¬L2Y v ¬L3B v L1W)
(¬L2B v ¬L3W v L1Y)
(¬L2B v ¬L3Y v L1W)

(¬O1W v C3Y)
(¬O1Y v C3W)
(¬O2Y v C1W)
(¬O2W v C1Y)
(¬O3Y v C2W)
(¬O3W v C2Y)

(¬O1Y v ¬O2W v O3Y v O3W) ⋀ (¬C3Y v ¬C3W v ¬C3B)
(¬O1Y v ¬O3W v O2Y v O2W) ⋀ (¬C2W v ¬C2Y v ¬C2B)
(¬O2Y v ¬O3W v O1Y v O1W) ⋀ (¬C1W v ¬C1Y v ¬C1B)

(¬C1W v ¬C2Y v C3B)
(¬C2W v ¬C3Y v C1B)
(¬C1W v ¬C3Y v C2B)
(¬C1W v ¬C2B v C3Y)
(¬C2W v ¬C3B v C1Y)
(¬C1W v ¬C3B v C2Y)
(¬C1B v ¬C2Y v C3W)
(¬C2B v ¬C3Y v C1W)
(¬C1B v ¬C2Y v C2W)

(¬O1Y v (C1Y v C1B))
(¬O1W v (C1W v C1B))
(¬O2Y v (C2Y v C2B))
(¬O2W v (C2W v C2B))
(¬O3Y v (C3Y v C3B))
(¬O3W v (C3W v C3B))
}

2d. ..

3. Can I get to work? (Forward Chaining)

Facts: { Rainy, HaveMountainBike, EnjoyPlayingSoccer, WorkForUniversity, WorkCloseToHome, HaveMoney, HertzClosed, AvisOpen, McDonaldsOpen }

1. e triggered: HaveMountainBike → HaveBike
    a. HaveBike added to consequences, {HaveBike}
2. m triggered: AvisOpen → CarRentalOpen
    a. CarRentalOpen added to consequences, {HaveBike, CarRentalOpen}
3. o triggered: CarRentalOpen → IsNotAHoliday
    a. IsNotAHoliday added to consequences, {HaveBike, CarRentalOpen, IsNotAHoliday}
4. k triggered: HaveMoney ^ CarRentalOpen → CanRentCar
    a. CanRentCar added to consequences, {HaveBike, CarRentalOpen, IsNotAHoliday, CanRentCar}
5. j triggered: CanRentCar → CanDriveToWork

        a. CanDriveToWork added to consequences, {HaveBike, CarRentalOpen, IsNotAHoliday, CanRentCar, CanDriveToWork}
6. b triggered: CanDriveToWork → CanGetToWork
    a. CanGetToWork added to consequences, {HaveBike, CarRentalOpen, IsNotAHoliday, CanRentCar, CanDriveToWork, CanGetToWork}
7. There are no more rules to be fully triggered, and therefore no more consequences. The final list of inferred propositions is as follows:
    a. {HaveBike, CarRentalOpen, IsNotAHoliday, CanRentCar, CanDriveToWork, CanGetToWork}

So, *yes, CanGetToWork* is among the final inferred propositions.

4. Backtracking

- Goal Stack: [CanGetToWork]
    - There are three rules that lead to CanGetToWork:
        - CanBikeToWork → CanGetToWork
        - CanDriveToWork → CanGetToWork
        - CanWalkToWork → CanGetToWork

    - Goal Stack: [CanBikeToWork]
        - Pop CanBikeToWork – all antecedents are valid.

    - Goal Stack: [CanDriveToWork]
        - Pop CanDriveToWork – there are two goals associated with it that must be pushed to the stack:

            - i. CanRentCar → CanDriveToWork
            - ii. HaveMoney ^ TaxiAvailable → CanDriveToWork

            - Goal Stack: [HaveMoney, TaxiAvailable]
                - Pop HaveMoney – it's valid. Now, I'll evaluate "TaxiAvailable."
                - Goal Stack: [TaxiAvailable]
                - TaxiAvailable is not a fact, so it's removed from the stack. Now, to evaluate i:
                - Push "CanRentCar" onto the stack and evaluate its antecedents:
                - Goal Stack: [HaveMoney, CarRentalOpen]
                - Pop "HaveMoney" – it's valid. Now, evaluate "CarRentalOpen."
                - Goal Stack: [CarRentalOpen]

        - There are three rules associated with "CarRentalOpen," so I now push the antecedents for these rules onto the stack:

        - Goal Stack: [HertzOpen, AvisOpen, EnterpriseOpen]
            - Pop "HertzOpen" – it's a fact. Now, you can continue with "AvisOpen" and "EnterpriseOpen."
            - Goal Stack: [AvisOpen, EnterpriseOpen]
            - AvisOpen and EnterpriseOpen are not facts, but I’ve already proven "CarRentalOpen" with "HertzOpen."

        - Goal Stack: []

    - The goal has been reached, and thus the stack is empty and CanGetToWork is proven through backtracking. No need to push CanWalkToWork.