Corrigé DM 5

Q1.
La phrase de la boîte 1 est $P_1 \vee P_2$.
Q2.
L'assertion 2 est $\neg P_1$.
Q3.
L'assertion du présentateur est $\big((P_1 \vee P2) \wedge \neg P_1\big) \vee \big(\neg(P_1 \vee P_2) \wedge \neg\neg P_1\big)$.
De par les règles de De Morgan, l'involutivité de $\neg$ et la distributivité de $\wedge$ sur $\vee$, cette assertion est équivalente à $\big((P_1 \wedge \neg P_1) \vee (P_2 \wedge \neg P_1)\big) \vee \big((\neg P_1 \wedge \neg P_2) \wedge P_1\big)$.
De par la commutativité et l'associativité de $\wedge$, la règle du tiers exclu ($P_1 \wedge \neg P_1 \Leftrightarrow F$ où F est la phrase toujours fausse) et le fait qu'une assertion fausse est absorbante pour $\wedge$ ($F \wedge P \Leftrightarrow F$) et neutre pour $\vee$ ($F \vee P \Leftrightarrow P$), cette assertion est équivalente à $P_2 \wedge \neg P_1$.
Q4.
Nous choisissons la boîte 2.
Q5.
L'assertion de la boîte 1 est $\neg P_1 \vee P_2$. L'assertion de la boîte 2 est P_1.
Q6.
L'assertion que les deux phrases sont simultanément vraies ou fausses est $\big((\neg P_1 \vee P_2) \wedge P_1\big) \vee (\neg(\neg P_1 \vee P_2) \wedge \neg P_1\big)$. Celle-ci est équivalente à $\big((\neg P_1 \wedge P_1) \vee (P_2 \wedge P_1)\big) \vee ((P_1 \wedge \neg P_2) \wedge \neg P_1\big)$ qui est équivalente à $P_2 \wedge P_1$. Les deux boîtes contiennent donc une clé verte. On peut choisir les deux boîtes pour gagner.
Q7.
Notons V_i l'assertion "la boite i est vide" et Q_i l'assertion "la boîte i contient la clé rouge". L'inscription 1 est V_3. L'inscription 2 est Q_1. L'inscription 3 est V_3.
Q8.
Cette assertion est $\big(V_3 \wedge Q_1 \wedge \neg V_3\big) \vee \big(V_3 \wedge \neg Q_1 \wedge V_3\big) \vee \big(\neg V_3 \wedge Q_1 \wedge V_3\big)$.
Q9.
Supposons que la clé verte est dans la boîte 2. Alors la clé rouge est dans la boîte 1. L'assertion 1 est donc fausse ; ainsi, la boîte 3 n'est pas vide. Cependant, la boîte 1 ayant une clé rouge et la boîte 2 une clé verte, la boîte 3 est vide. Ceci est contradictoire. Par conséquent, la boîte 2 ne contient pas la clé verte.
Q10.
Supposons que la clé verte est dans la boîte 3. Alors l'assertion de cette boîte est vraie donc cette boîte est vide, ce qui est absurde. La clé verte n'est donc ni dans la boîte 2 ni dans la boîte 3. Elle est donc dans la boîte 1. L'assertion 1 est donc vraie ; la boîte 3 est donc vide. Enfin, la clé rouge ne peut être que dans la boîte 2.
En conclusion, la boîte 1 contient la clé verte, la boîte 2 la clé rouge et la boîte 3 est vide.

Q12.

```
let c = Ou ( Ou ( Var(0),Var(1)), Non( Var(2) ));;
```

Q13.

```
let f = [ Ou ( Ou ( Var(0), Var(1)), Non( Var(2))) ; Ou (Non ( Var(1)), Var(2))];;
```

Q14.

```
let rec evalue_clause c t = match c with
    | Var(i) -> t.(i)
    | Non(c0) -> not (evalue_clause c0 t)
    | Ou (c1,c2) -> evalue_clause c1 t || evalue_clause c2 t;;
```

Q15.

```
let rec evalue_FNC f t = match f with
    | [] -> true
    | c::f0 -> (evalue_clause c t) && (evalue_FNC f0 t);;
```

Q16. On obtient "true".
Q17.

```
let suivant t = let n=Array.length t in let rec aux i r =
   if i=0 then not r
   else
       if not(r) then true
       else begin
           t.(i) <- not ( t.(i));
           aux (i-1) not(t.(i))
       end in aux (n-1) true;;
```

Q18.

```
let satisfiable f n =let t=Array.make n false in
    let rec aux tab = (evalue_FNC f tab) || (suivant tab && (aux tab));;
```

Q19. L'évaluation d'une "FNC" demande une complexité de l'ordre de la taille (que l'on notera m) de cette "FNC". Le passage à la table suivante demande une complexité de l'ordre de n. Il y a 2^n tables de vérité. Au total, la complexité est donc $O(mn2^n)$.

Q20. Soit $f = c_1 \wedge \cdots \wedge c_n$ une FNC. Pour la satisfaire, il faut (et il suffit) que chaque c_i soit satisfaite. Pour chaque $c_i = (\bigvee_{j\in I_i} x_j) \vee (\bigvee_{k\in J_i} \neg x_k)$, il faut (et il suffit) que l'un x_j, pour $j \in I_1$ soit vrai ou que l'un des x_k, pour $k \in I_2$, soit faux.

On va donc, pour chaque clause c_i, tenter pour chaque $j \in I_i$, essayer de trouver une solution pour laquelle $x_j = VRAI$ ou, pour chaque $k \in J_i$, essayer de trouver une solution pour laquelle $x_k = FAUX$. En cas d'échec à trouver une telle solution, on réinitialisera x_j ou x_k (sans lui donner de valeur) et on continuera le procédé.

L'algorithme est donc le suivant :

t :=tableau de taille N, le nombre de variables, de valeurs "None" (ou plutôt "unit" en Ocaml).

fonction tenter (i)= si i=N+1, renvoyer t sinon pour chaque j dans I_j faire si t.(j) n'a pas de valeur faire t.(j) :=VRAI tenter(i+1) t.(j) :=None pour chaque k dans J_i faire si t.(k) n'a pas de valeur faire t.(k) :=FAUX tenter(i+1)

t.(k) :=None **Q21.**

F := concaténation(F1,...,Fn, not (Phi)) renvoyer not (satisfiable (F))