Corrigé DS2

Problème 1

1. Ce terme est un (string, char) terme qui représente le terme $x + \cos(y)$.

2. Il suffit de vérifier pour chacun des symboles de fonctions si la taille de la liste de ses arguments est égale à son arité. On peut exploiter pour se faire List.for_all :

```
let rec terme_bien_forme t :bool = match t with
  |V(_) -> true
  |F((_, arite), l) -> (List.length l = arite) && List.for_all terme_bien_forme l
```

3. On utilise le terme défini en question 1 :

```
let f = Forall('x', R(("egal",2),[t;t]))
```

4. Même principe que pour la question 3. On prend garde de vérifier la correction des arités des fonctions utilisées dans chacun des termes dans le cas d'une formule atomique :

```
let rec formule_bien_formee f = match f with
  |R((_,arite),l) -> List.length l = arite && List.for_all terme_bien_forme l
  |Forall(_,g) |Exists(_,g) -> formule_bien_formee g
  |Non(g) -> formule_bien_formee g
  |Op_bin(_,g,h) -> (formule_bien_formee g) && (formule_bien_formee h)
```

5. On peut exploiter List.map conjointement avec List.mem ou utiliser directement List.exists :

```
let rec apparait x t = match t with
  |V(y) -> x=y
  |F(_,l) -> List.mem true (List.map (apparait x) l)
(* ou : List.exists (apparait x) l *)
```

6. Une occurrence de variable est libre si elle n'est sous la portée d'aucun quantificateur. On obtient :

```
let rec est_libre x f = match f with
  |R(_,l) -> List.exists (apparait x) l
  |Forall(y,g) |Exists(y,g) -> x<>y && (est_libre x g)
  |Non(g) -> est_libre x g
  |Op_bin(_,g,h) -> est_libre x g || est_libre x h
```

7. Une formule est close si toutes les occurrences de chacun de ses variables est liée. On applique donc pour chacune des variables v le traitement suivant : vérifier si une occurence de v est libre à l'aide de est_libre. Si ce n'est le cas pour aucune variable, on renvoie true et sinon false.

```
let est_close f lv =
  not (List.exists (fun v -> est_libre v f) lv)
```

8. La structure $\mathcal{M}$ n'est pas un modèle de F car ce n'est pas un modèle de F_1 : en effet, $1 < 1$ est faux. En revanche, en modifiant $\mathcal{M}$ de sorte à ce que R soit interprétée par $\leq$ sur les réels, on obtient un modèle $\mathcal{M}'$ pour F. En fait, n'importe quel ensemble ordonné est un modèle pour F.

9. a) Ce séquent est valide et on peut même remplacer l'implication par une équivalence et conserver ce fait. On l'a démontré en cours dans le cadre du calcul propositionnel mais rien ne change dans le cadre du calcul des prédicats. On note $\Gamma = \{F_1 \vee F_2, \neg F_1 \wedge \neg F_2\}$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, F_1 \vdash \neg F_1 \wedge \neg F_2}\,\text{ax}}{\Gamma, F_1 \vdash \neg F_1}\,\text{elim-}\wedge \quad \dfrac{}{\Gamma, F_1 \vdash F_1}\,\text{ax}}{\Gamma, F_1 \vdash \bot}\,\text{elim-}\neg \quad \dfrac{\dfrac{\dfrac{}{\Gamma, F_2 \vdash \neg F_1 \wedge \neg F_2}\,\text{ax}}{\Gamma, F_2 \vdash \neg F_2}\,\text{elim-}\wedge \quad \dfrac{}{\Gamma, F_2 \vdash F_2}\,\text{ax}}{\Gamma, F_2 \vdash \bot}\,\text{elim-}\neg \quad \dfrac{}{\Gamma \vdash F_1 \vee F_2}\,\text{ax}}{\dfrac{\dfrac{\Gamma \vdash \bot}{\Gamma \vdash \neg(F_1 \vee F_2)}\,\text{intro-}\neg}{\vdash \neg F_1 \wedge \neg F_2 \Rightarrow \neg(F_1 \vee F_2)}\,\text{intro-}\Rightarrow}\,\text{elim-}\vee$$

b) Ce séquent n'est pas valide car il existe des ensembles ordonnés non totalement ordonnés. Par exemple, la structure dont le domaine est $\{a, b\}^\star$ et dans laquelle R est interprétée par l'ordre préfixe sur les mots satisfait F mais ne satisfait pas $\forall x \, \forall y \, R(x, y) \vee R(y, x)$ puisque ab et ba ne sont pas comparables.

c) Ce séquent est valide comme le montre l'arbre de preuve suivant :

$$\dfrac{\dfrac{\dfrac{\dfrac{}{F \vdash F}\,\text{ax}}{F \vdash \forall x \, R(x, x)}\,\text{elim-}\forall\ (2\text{ fois})}{F \vdash R(x, x)}\,\text{intro-}\exists}{F \vdash \exists y \, R(x, y)}\,\text{intro-}\forall$$

L'intro-$\exists$ peut être utilisée car $R(x, x) = R(x, y)[x/y]$. L'utilisation de la règle intro-$\forall$ est licite car x n'est pas libre dans F. Cette preuve formalise le raisonnement sémantique suivant : si on a F, on a toujours $R(x, x)$ donc en particulier, il existe un y tel que $R(x, y)$, à savoir x lui même.

d) Ce séquent n'est pas valide car il existe des ordres bien fondés. Par exemple, la structure dont le domaine est $\mathbb{N}$ et dans laquelle R est interprétée par $\geq$ est un modèle de F mais pas un modèle de $\forall x \, \exists y \, (R(x, y) \wedge \neg(x = y))$. En effet, il n'y a pas d'entier naturel y tel que $0 \geq y$ et $y \neq 0$.

e) Ce séquent n'est pas valide car dans il existe des ensembles ordonnés qui n'admettent pas de majorant. La structure $\mathcal{M}'$ définie à la question 8 est un modèle pour F mais pas pour $\exists x \, \forall y \, R(x, y)$.

Problème 2

1. Il y en a autant que le cardinal de A^n, c'est-à-dire p^n.

2. Il y a autant de mots de longueur n formés de lettres toutes distinctes qu'il n'y a d'applications injectives de $[\![1, n]\!]$ dans A, c'est-à-dire 0 si $n > \text{Card}(A) = p$ et $n(n-1)...(n-p+1) = \frac{n!}{(n-p)!}$ si $p \geq n$.

3. Si n est pair, il existe $k \in \mathbb{N}$ tel que $n = 2k$ et dans ce cas un palindrome de taille n est entièrement déterminé par ses k premières lettres : il y a donc $p^k = p^{n/2}$ palindromes de taille n paire, d'après la question 1.

   Si $n = 2k+1$ est impair, un palindrome de taille n est entièrement déterminé par ses $k+1$ premières lettres : il y a donc $p^{k+1} = p^{(n+1)/2}$ palindromes de taille n impaire.

4. On note E_n l'ensemble des mots de A^n n'ayant jamais deux lettres consécutives égales et C_n son cardinal. Si $p = 1$, on a $C_1 = 1$ et pour tout $n \geq 2$, $C_n = 0$ car sur un alphabet à une lettre, un mot d'au moins deux lettres a forcément deux lettres consécutives égales. On se place donc dans le cas où $p \geq 2$.

   Montrons alors par récurrence sur $n \in \mathbb{N}^\star$ la propriété $H(n)$ suivante : $C_n = p(p-1)^{n-1}$.