Examen - Septembre 2008

Durée : 3h

Les notes de cours, de travaux dirigés et de travaux pratiques sont autorisées. Le sujet comporte deux pages et trois parties. Le barème est donné à titre indicatif. Vous vous attacherez à soigner la présentation générale et l'orthographe, et vous serez notamment vigilant quant à la syntaxe lors de l'écriture de fragment de code devant être accepté par le système Coq.

1 Questions de cours (5 pts)

Question 1 On rappelle les types de quelques constantes :

```
or_ind : forall A B P : Prop, (A -> P) -> (B -> P) -> A \/ B -> P
or_intror : forall A B : Prop, B -> A \/ B
or_introl : forall A B : Prop, A -> A \/ B
conj : forall A B : Prop, A -> B -> A /\ B
and_ind : forall A B P : Prop, (A -> B -> P) -> A /\ B -> P
```

Précisez les énoncés logiques correspondant aux types des trois termes de preuve suivants :

```
fun (A : Prop) (H : A) (H0 : ~ A) => H0 H
```

```
fun (A B : Prop) (C : Type) (X : A /\ B -> C) (H : A -> B) (H0 : A) =>
X (conj H0 (H H0))
```

```
fun (A B C : Prop) (H : A /\ (B \/ C)) =>
and_ind
  (fun (HA : A) (HBC : B \/ C) =>
   or_ind (fun HB : B => or_introl (A /\ C) (conj HA HB))
     (fun HC : C => or_intror (A /\ B) (conj HA HC)) HBC) H
```

Question 2 On reprend la définition usuelle des entiers naturels :

```
Inductive nat : Set := O : nat | S : nat -> nat.
```

On considère la fonction suivante :

```
Fixpoint modulo3 (a : nat) :=
match a with
| O => O
| S O => 1
| S (S O) => 2
| S (S (S p)) => modulo3 p
end.
```

Ecrivez les règles de calcul associées à cette définition. On rappelle que les règles demandées sont celles qui s'appliquent lors de l'appel à la tactique `simpl`.

2 Itérateurs (8 pts)

On considère le type des listes d'entiers :

```
Inductive list : Set :=
| nil : list
| cons : nat -> list -> list.
```

On va chercher à construire des itérateurs pour faire des traitements sur les listes.

On commence par redéfinir la fonction `map` de la manière suivante :

```
Fixpoint map (f:nat->nat) (l:list) : list :=
match l with
| nil => nil
| cons x xs => cons (f x) (map f xs)
end.
```

Question 3 Ecrivez le principe d'induction associée à la définition inductive `list`.
Question 4 Quel est le type de la fonction `map` ?
Question 5 On va maintenant définir une fonction `apply` de type

$$\texttt{nat} \to (\texttt{nat} \to \texttt{nat} \to \texttt{nat}) \to \texttt{list} \to \texttt{nat}.$$

La fonction `apply` permet d'appliquer une fonction binaire sur une liste d'arguments et donc de la rendre n-aire. Ainsi `apply zero plus` $[1, 2, 3]$ renvoie la somme des éléments de la liste, à savoir 6. Programmez la fonction `apply` par point-fixe et filtrage.
Question 6 On cherche maintenant à programmer une fonction `enum` qui construit la liste ordonnée des entiers naturels de 0 à n. Cette fonction a pour type $\texttt{nat} \to \texttt{list}$. Vous êtes libre de construire la liste dans l'ordre croissant ou décroissant. Justifiez simplement quel sens rend la programmation plus simple. Programmez cette fonction dans `Coq`.
Question 7 A partir des fonctions précédentes, construisez une fonction qui calcule $\Sigma_{i=0}^{n}\ i^2$ à partir d'un entier n donné en argument.

3 Parité (7 pts)

On considère une définition mutuellement inductive des propriétés *être pair* et *être impair*.

```
Inductive even : nat -> Prop :=
| even_O : even O
| even_S : forall n : nat, odd n -> even (S n)
with odd : nat -> Prop :=
| odd_S : forall n : nat, even n -> odd (S n).
```

Question 8 Montrez comment, en `Coq`, on peut démontrer la propriété `odd (S O)` grâce à cette définition.
Question 9 Expliquez comment prouver en Coq (écrire les tactiques) la propriété suivante

$$\forall n : \texttt{nat},\ \texttt{even}\ n \to \texttt{even}\ (\texttt{S}\ (\texttt{S}\ n)).$$

Question 10 On reprend la notion de liste présentée dans la partie précédente. Programmez une fonction qui réutilise `map` et `apply` et teste si tous les entiers de la liste fournie en argument sont pairs. Sur quel type d'objets la fonction `apply` doit-elle s'appliquer ? Le type de la fonction `map` convient-il pour cette application ? Sinon, comment faudrait-il modifier la fonction et son type ?
Question 11 Programmez une seconde fonction qui teste si au moins un des entiers de la liste est pair. Quelles résultats attend-t-on de ces fonctions (celles des questions 10 et 11) sur la liste vide ?