Contrôle Continu - 3 décembre 1999

Les seuls documents autorisés sont les documents manuscrits et les documents distribués en cours, TDs et TPs. La durée de l'épreuve est de 2h.

Exercice 1 (10 points)

Vous pouvez utiliser toutes les fonctions vues en cours ou appartenant aux bibliothèques décrites dans le manuel de référence de Caml.

Un permis de conduire français contient, entre autres informations, l'identité de son possesseur et la liste des catégories de permis de cette personne.

Les catégories de permis considérés ici sont `A` (permis moto) et `B` (voiture).

Une personne est identifiée par son nom et sa ville donnée par son code postal.

1.1 Définir les types `catégorie_permis`, `identité` et `permis`.

Par la suite, nous considérerons qu'une personne qui n'a pas de permis de conduire dispose d'un élément de type `permis`, mais que la liste des catégories de permis qu'il possède est vide.

1.2 Écrire la fonction `permis_voiture` qui détermine si la personne détentrice d'un permis `p` peut conduire une voiture.

Écrire la fonction `nouveau_permis` qui met à jour le permis `p` d'une personne qui vient d'être reçue à l'examen du permis de conduire pour la catégorie `c`.

1.3 Sachant que le département se déduit du code postal en divisant ce dernier par 1000, écrire la fonction `département_of_ville` qui indique le département associé à la ville `v`.

1.4 À la suite d'un contrôle routier, un policier a relevé une liste de permis de contrevenants, il souhaite classer ces permis selon l'ordre alphabétique des noms de ces personnes. Écrire la fonction `tri_alph_nom` qui trie une liste de permis selon l'ordre alphabétique des noms de leurs détenteurs. Vous pourrez pour cela utiliser la fonction `tri_insertion` vue en cours ou bien la fonction `sort` du module `sort` qui fait partie de la bibliothèque standard de Caml.

1.5 Devant la variété d'origine de ces détenteurs de permis, ce policier décide de les classer d'abord par ordre de département et pour chaque département en ordonnant les noms des contrevenants. Écrire la fonction `tri_plus_complexe` associée.

1.6 Écrire une fonction `détenteurs_permis_voiture` qui extrait d'une liste de permis ceux qui permettent de conduire une voiture. On écrira cette fonction de deux façons différentes, tout d'abord directement récursivement, puis en utilisant l'itérateur `filter` vu en cours.

Exercice 2 (7 points)

2.1 Typez complètement les déclarations suivantes. Une réponse ne comportant que le type de la fonction ne sera pas considérée comme une réponse correcte.

La session débute dans l'environnement de typage global `E0`. Les environnements seront ensuite nommés `E1`, `E2`,...

```
let x = 1;;
let f1 = function y -> y + x;;
let f2 =
  function x -> function y ->
    if x > y then x+y else f1(x-y);;
let deriv f dx x = (f(x +. dx) -. f(x)) /. dx;;
let partielle = deriv (function x -> x *. x) 0.001;;
```

2.2 Donnez, sans détailler, le type de cette fonction et de son application :

```
let rec iter f n =
  if n = 0 then 1 else f (iter f (n-1));;
iter f1 1000;;
```

Exercice 3 (7 points)

3.1 Typez complètement chacune de ces déclarations en montrant également l'évolution de l'environnement. On supposera que `fst : 'a * 'b -> 'a` et `snd : 'a * 'b -> 'b`.

```
let copie x = (x,x);;
let f (g,x) = g(snd x),g(fst x);;
let h x = f (copie,x);;
let m x = h (x+1,x-1);;
```

3.2 Expliquez la différence (du point de vue du typage) entre les deux expressions suivantes :

```
let id = function x -> x;;
if id true then id 1 else id 2;;
```

et :

```
let id = function x -> x;;
(function f -> if f true then f 1 else f 2) id;;
```