# Algorithmique
# Contrôle n° 1

INFO-SUP S1
EPITA

24 Oct 2016 - 09 h 00 (1 h)

**Remarques (à lire !) :**

- Vous devez répondre sur **les feuilles de réponses prévues à cet effet**. Aucune autre feuille ne sera ramassée (gardez vos brouillons pour vous).
  Ne séparez pas les feuilles à moins de pouvoir les réagrafer pour les rendre.
- La présentation est notée en moins, c'est à dire que vous êtes noté sur 20 et que vos points de présentation (2 au maximum) peuvent être retirés de cette note.
- Tout code CAML non indenté ne sera pas corrigé.
- Tout code CAML doit être suivi de son évaluation : la réponse de CAML.
- En l'absence d'indication dans l'énoncé, les seules fonctions que vous pouvez utiliser sont `failwith` et `invalid_arg` (aucune autre fonction prédéfinie de CAML).
- Aucune réponse au crayon de papier ne sera corrigée.
- Durée : 2h00 (May the force...)

## Exercice 1 (Types Abstraits : Ensemble (Nombre d'éléments) – *5 points*)

Dans le cas des structures séquentielles étudiées en cours, la notion d'ordre des éléments appartenant à une liste est fondamentale. Ce n'est pas toujours le cas : l'ordre des éléments peut n'avoir aucune signification et la propriété importante est la présence ou l'absence d'un élément.

On retrouve alors la notion d'***ensemble d'éléments*** : On veut pouvoir tester l'appartenance d'un élément à un ensemble, ajouter ou supprimer un élément, tester si un ensemble est vide, etc.

Dans ce cas, la signature du type ***ensemble*** comporte au moins les types et opérations suivantes :

**TYPES**

ensemble

**UTILISE**

élément, booléen

**OPÉRATIONS**

| | | |
|---|---|---|
| ensemble-vide : | → ensemble | |
| ajouter | : | élément × ensemble → ensemble |
| supprimer | : | élément × ensemble → ensemble |
| _ ∈ _ | : | élément × ensemble → booléen |

Dans ce qui suit, $x$ et $y$ sont des variables de type *élément* et $e$ est une variable de type *ensemble*.

**AXIOMES**

$x \in$ ensemble-vide = faux
$(x = y) \Rightarrow (x \in \text{ajouter}(y, e)) = \text{vrai}$
$(x \neq y) \Rightarrow (x \in \text{ajouter}(y, e)) = (x \in e)$
$(x = y) \Rightarrow (x \in \text{supprimer}(y, e)) = \text{faux}$
$(x \neq y) \Rightarrow (x \in \text{supprimer}(y, e)) = (x \in e)$

**AVEC**

ensemble e
élément x, y

Supposons l'opération *card* déterminant le nombre d'éléments d'un ensemble, dont le nom et le profil sont :

**OPÉRATIONS**

card : ensemble → entier

Donner les axiomes qui définissent suffisamment cette opération.

**Exercice 2 (Type abstrait *Liste récursive* → liste CAML – *5 points*)**

On étend le type abstrait *liste récursive* vu en cours en lui ajoutant l'opération *mystère* :

**TYPES**
liste

**UTILISE**
élément

**OPÉRATIONS**

*mystère* : élément × liste → liste

**AXIOMES**

$mystère(x, listevide) = cons(x, listevide)$
$x <= e \Rightarrow mystère(x, cons(e, \lambda)) = cons(x, cons(e, \lambda))$
$x > e \Rightarrow mystère(x, cons(e, \lambda)) = cons(e, mystère(x, \lambda))$

**AVEC**

$\lambda$ : liste
e : élément

En CAML , on implémente, naturellement, les *listes récursives* par des `'a list` (et donc *élément* est le type `'a`).

1. Donner les spécifications (paramètres, ce qu'elle fait) de la fonction CAML `mystery` implémentant l'opération *mystère*.

2. Écrire la fonction `mystery` respectant vos spécifications.

---

**Exercice 3 (Combien ? – *4 points*)**

1. Écrire la fonction CAML `how_many` dont les spécifications sont les suivantes :
   — Elle prend en paramètre une fonction booléenne $f$ ainsi qu'une liste : $[a_1; a_2; \cdots; a_n]$.
   — Elle recherche dans la liste les valeurs $a_i$ telle que $f(a_i)$ sont vrai et retourne le nombre de valeurs trouvées.

2. Utiliser la fonction `how_many` pour définir une fonction qui retourne le nombre de valeurs multiples d'un entier $n$ donné dans une liste d'entiers.

---

**Exercice 4 (exists2 – *6 points*)**

1. Écrire la fonction CAML `exists2` dont les spécifications sont les suivantes :
   — Elle prend en paramètre une fonction booléenne à deux paramètres : `p` ainsi que deux listes : $[a_1; a_2; \cdots; a_n]$ et $[b_1; b_2; \cdots; b_n]$.
   — Elle retourne le booléen : il existe au moins un couple $(a_i, b_i)$ tel que $p\ a_i\ b_i$ est vrai.
   — Elle déclenche une exception si les deux listes sont de longueurs différentes (si aucun couple $(a_i, b_i)$ tel que $p\ a_i\ b_i$ est vrai n'a été trouvé...).

2. Utiliser la fonction `exists2` pour définir une fonction qui vérifie si deux listes sont identiques.