Mise à Niveau en Caml

Année 2013-2014

Types de base et Traitement des listes

Durée : 4 h environ

L'objet de cette mise à niveau est de vous introduire les bases du langage CAML que nous utiliserons en TRP. Nous commençons par écrire quelques expressions dans les différents types de base puis des fonctions simples manipulant des listes et des structures de données utilisant des listes.

Les projets sont à réaliser en *Objective Caml*.

– Le manuel de référence du langage est disponible en ligne à l'adresse suivante : **http://caml.inria.fr/pub/docs/manual-ocaml**.

– Le langage CAML peut être téléchargé à l'adresse : **http://caml.inria.fr/download.fr.html**.

– Un manuel d'initiation à OCaml est disponible à l'adresse suivante : http://www-igm.univ-mlv.fr/~beal/Enseignement/Logique/ocaml.pdf.

Mise en route

- Logez-vous sur "ouvea".
- Depuis le terminal, lancez un éditeur :
  - nedit debutCaml.ml &
  - ou emacs debutCaml.ml &
  - ou xterm & puis vi debutCaml.ml
- Appelez CAML en tapant la commande « **ocaml** » dans un shell. (Vous pouvez également lancer ocaml depuis Emacs.) Vous obtenez :

```
                Objective Caml version 3.10.2

#
```

- On peut ensuite taper des instructions directement, une à une, ou bien les écrire dans un fichier puis charger ce fichier en tapant l'instruction « **#use "fichier.ml" ;;** ».
- On **sort** de l'interpréteur CAML en tapant l'instruction « **exit 0 ;;** ».

Rappels sur les types

1. Tapez une constante de type **bool**, (faîtes de même pour **int, char, float**)
2. Tapez une expression avec au moins un opérateur et renvoyant un résultat de type **bool** (idem pour **int, char, float**).
3. Tapez une constante de type **string**, puis une expression de type **string** contenant au moins un opérateur.
4. Tapez une expressions de type **unit**.
5. Attribuez à la variable **a** une expression de type **bool*int*int**.
6. Écrire une expression de type **int->int->int**.

Fonctions à écrire :

1. Écrire la fonction **est_pair** de type **int->bool**.
2. Écrire la fonction **max** de 3 éléments de même type (quelconque) : **'a->'a->'a->'a**.
3. Écrire la fonction **syracuse** qui admet un seul argument de type entier, elle renvoie son argument divisé par 2 s'il est pair et renvoie (3 fois son argument) plus 1 sinon.
4. Écrire la fonction **factorielle**.

5. Ré-écrire la fonction **`est_pair`** récursivement sans utiliser le chiffre 2. Expliquez les résultats obtenus avec 100 millions et -6.
6. Écrire la fonction qui affiche (utiliser **`print_int`**) les résultats des applications successives de la fonction **`syracuse`** à un nombre **`n`**, puis au résultat obtenu, etc., jusqu'à atteindre le nombre 1, qu'elle affiche également.
7. Écrire une fonction qui inverse une chaîne de caractères (utiliser **`String.sub`** et **`String.length`**).

Rappels sur les listes

La liste est la structure de donnée récursive la plus utilisée en CAML. La librairie « **`List`** » contient de nombreuses fonctions manipulant les listes ; elle est documentée à l'adresse « **`http://caml.inria.fr/pub/docs/manual-ocaml/libref/List.html`** » (pour accéder à une fonction de cette librairie, on fait précéder le nom de la fonction du préfixe **`List.`**)

Écrire en CAML, sans utiliser la librairie, les fonctions suivantes :

1. **`insererEnTete`** qui insère un élément **`x`** de type quelconque en tête d'une liste **`l`** et renvoie la nouvelle liste (vérifiez que le type de cette fonction est bien **`'a->'a list ->'a list`**).

2. **`insererEnQueue`** qui insère un élément de type quelconque en queue d'une liste et renvoie la nouvelle liste ;

3. **`somme`** qui retourne la somme de tous les éléments d'une liste d'entiers ;

4. **`maximum`** qui retourne la valeur maximale contenue dans une liste d'entiers ;

5. **`insererEntierDansListeOrdonnee`** qui insère un entier au bon endroit dans une liste ordonnée d'entiers triés par valeurs croissantes.

6. **`filtre : 'a list ('a bool) 'a list`** qui prend une liste et une fonction en paramètres et renvoie la liste dont les éléments correspondent au critère donné par la fonction (on pourra tester avec la fonction **`est_pair`** sur une liste d'entiers)

7. **premier : 'a list → ('a * 'a → bool) → 'a** qui prend une liste et une fonction en paramètres et renvoie l'élément minimum de la liste selon l'ordre donné par la fonction (on pourra tester avec la fonction **`inferieur`** qui pour un couple (x,y) renvoie vrai si x<y et faux sinon)

Listes d'associations

On s'intéresse maintenant à des listes de paires **`(n,x)`**, où **`n`** est un entier et **`x`** est de type quelconque **`'a`**, ces listes sont appelées *listes d'associations*. Écrire en CAML les fonctions suivantes:

8. **`minimumA`** qui retourne l'objet associé à l'entier minimal d'une liste d'associations ; vous pourrez au préalable écrire la fonction **`minimumArec`** qui retourne l'association dont l'entier est le plus petit.

Testez la fonction **`minimumA`** dans la liste suivante : **`[(9,'b');(3,'z');(8,'a');(4,'p')]`**.

9. **`supprime`** qui supprime l'association qui contient un objet donné dans une liste d'associations.

10. **`insererA`**, qui insère une association **`(n,x)`** dans une liste d'associations ordonnées selon les entiers par valeurs croissantes ; s'il y avait déjà une occurrence de **`x`** associée à un entier **`m`**, il doit n'en rester qu'une associée au minimum de **`m`** et **`n`** ; par exemple :

```
insererA (4,'b') [ (3,'z'); (4,'p'); (8,'a'); (9,'b') ] ;;
- : (int * char) list = [(3, 'z'); (4, 'b'); (4, 'p'); (8, 'a')]
```