MPIL(LI332)
Année 2014/2015
Séance 2 de TD/TME

Objectifs

1. Programmation fonctionnelle.
2. Filtrage par motifs.
3. Listes.
4. Définition de types somme.

Travaux Dirigés

Comme pour la séance précédente, et toutes les autres, n'oubliez pas de donner le type de toutes les fonctions que vous écrirez.

Exercice 1 : Filtrage

Q.1.1 La fonction calculant le `n`-ième nombre de la suite de Fibonacci peut être ainsi définie pour tout nombre entier positif :

$$\begin{cases} fib(0) & = & 1 \\ fib(1) & = & 1 \\ fib(n) & = & fib(n-2) + fib(n-1) \end{cases}$$

Écrivez cette fonction en Caml en utilisant un filtrage.

Q.1.2 (Quiz) Dites si les codes suivant compilent et s'ils se comportent comme ce que leur auteur a voulu écrire. Si non, corrigez-les.

1. ```
   let egale_trois = function
       3 -> true
     | _ -> false
   ```

2. ```
   let egale_ixe x = function
       x -> true
     | _ -> false
   ```

3. ```
   let double_zero n m =
     match n m with
       | 0 0 -> true
       | _ -> false
   ```

4. ```
   let rec sorted = function
       [] -> true
     | x :: y :: q -> x < y && sorted q
   ```

5. ```
   let rec intercale l sep =
     match l with
       | s :: l -> s ^ sep ^ intercale l sep
       | s :: [] -> s
       | [] -> ""
   ```

Exercice 2 : Sommes et filtrage

Q.2.1 À la petite école, vous avez vu la représentation dite *unaire* des nombres entiers : pour un nombre entier n donné, on le représente en posant exactement n bâtons sur la table. Les entiers de *Peano* formalisent cette représentation intuitive des nombres entiers : un nombre n est soit zero (aucun bâton sur la table) soit le successeur d'un autre nombre n' (on ajoute un bâton aux bâtons nécessaires pour représenter n').

Définissez sous forme de type Caml cette représentation des entiers.

Q.2.2 Écrivez les fonctions de conversion vers les entiers machine prédéfinis dans OCaml : `peano_of_int` et `int_of_peano`.

Q.2.3 Donnez une méthode pour tester que la conversion est valide jusqu'à un certain rang et écrivez une fonction inplantant cette méthode.

Q.2.4 Écrivez le prédicat `even` décidant si un nombre en représentation de Peano est pair, sans passer par la conversion. **Rappel :** le filtrage peut parcourir en profondeur les valeurs.

Q.2.5 Vérifiez que le prédicat est valide, de la même façon que la conversion.

Q.2.6 Déduisez une manière de tester qu'une fonction agissant que les entiers de Peano se comporte de la même façon que sa version travaillant sur les entiers OCaml standard. Écrivez une telle fonction de test générique et donnez son type.

Q.2.7 Écrivez les fonction `half` et `twice` et vérifiez-les.

Q.2.8 Testez les fonctions half et twice ainsi que les précédentes en utilisant la fonction de test générique.

Exercice 3 : Tris par insertion

On veut réaliser une fonction de tri par insertion pour les listes.

Q.3.1 Écrire une fonction `ins` prenant un entier `x` et une liste d'entiers `l` triée en paramètre et insérant `x` dans `l` de manière telle que la liste obtenue soit triée.

Q.3.2 En déduire la fonction de tri par insertion `sort`.

Q.3.3 Quelle est la complexité de ce tri ?

Travaux sur Machines Encadrés

Exercice 4 : Listes

On veut créer une fonction générant, pour un entier $k > 0$, la suite de k lignes suivantes :

```
liste 1: [1]
liste 2: [1;1]
liste 3: [2;1]
liste 4: [1;2;1;1]
liste 5: [1;1;1;2;2;1]
liste 6: [3;1;2;2;1;1]
```

On obtient la liste `k` en lisant le contenu de la liste $k-1$.

Q.4.1 Écrire une fonction `calcul_prefixe` qui prend une liste `l` ayant la forme des listes mentionnées ci-dessus, et qui renvoie 0 si `l` est vide ou le nombre de chiffres identiques au début de la liste `l`. Par exemple, l'application de `calc_prefixe` sur la liste 5 ci-dessus renvoie 3.

Q.4.2 Écrire une fonction `genere_liste` qui prend en argument une liste ayant la forme des listes mentionnées ci-dessus et qui renvoie la liste suivante.

Q.4.3 Écrire une fonction `genere` qui prend en argument un entier $k > 0$ et qui renvoie une liste contenant les k premières listes de la suite.

Exercice 5 : Manipuler des chaînes de caractères

Comme nous l'avons déjà vu lors de la séance précédente, on peut concaténer deux chaînes de caractères avec l'opérateur ^ directement mais la plupart des fonc-
```

tions disponibles permettant de manipuler des chaînes de caractères se trouvent dans le module `String` (documentation à l'adresse : http ://caml.inria.fr/pub/docs/manual-ocaml/libref/String.html).

Consulter la description des fonctions se trouvant dans le module `String` dans le manuel de référence. Essayez les fonctions `length`, `get`, `index`, `sub`.

Q.5.1 Écrire une fonction `cherche_car` qui cherche si un caractère `car` apparaît dans une chaîne de caractères `str` passée en paramètre : si tel est le cas, elle renvoie la position de `car` dans `str`, sinon elle renvoie -1.

Q.5.2 Étant donnée une chaîne `str`, on veut modifier `str` en ajoutant un espace derrière chaque point et chaque virgule apparaissant dans `str`. Écrire une fonction OCaml qui remplit cette tâche.

Exercice 6 : Jeux de Belote

Q.6.1 Définir un type `couleur` pour représenter les couleurs d'un jeu de cartes.

Q.6.2 Définir un type `carte` pour représenter les cartes d'un jeu de belote : as, roi, dame, valet et petites cartes.

Q.6.3 Écrire une fonction `valeur` qui prend la couleur de l'atout et une carte et renvoie la valeur de la carte. On rappelle les règles de comptage suivantes : l'as vaut 11, le roi 4, la dame 3, le valet 20 s'il a la couleur de l'atout, 2 sinon, le 10 vaut 10, le 9 vaut 14 s'il a la couleur de l'atout, 0 sinon, les autres cartes valent 0.

Q.6.4 Tester la fonction précédente en déclarant un nombre `mon_jeu` qui calcule la valeur du jeu suivant : valet de carreau, 10 de trèfle, 7 de cœur, 8 de carreau, 9 de pique avec atout à carreau (on doit trouver 30) .