NFP 119 : Programmation Fonctionnelle

Session de Septembre 2009

septembre 2009

Il vous est recommandé de rédiger de façon concise et précise, tout en écrivant lisiblement. Il sera tenu compte de la présentation dans la note de la copie. Tous les documents sont autorisés. Le barème n'est donné qu'à titre indicatif

Exercice 1 (6 points):

On veut modéliser, à l'aide des listes, l'indexation d'un ensemble de documents selon leurs mot-clés. Dans cet index, chaque document est représenté par une paire $(t, lMots)$, où t est son titre, et $lMots$ est sa liste de mots-clés.

```
type index = (string * string list) list;;

let i = [("Amon-Re",["Egypte"; "dieux"]);
       ("Voyager␣en␣Egypte",["Egypte"; "voyages"]);
       ("Typeful␣Programming", ["typage"; "programmation";"informatique"]);
       ("Java␣pour␣les␣nuls", ["programmation";"informatique";"Java"])];;
```

Afin d'exploiter ces index vous devez écrire les fonctions de recherche suivantes :

1. la fonction `chercheUnMot` qui prend un index i et un mot-clé m, et qui renvoie la liste de tous les titres (et uniquement les titres) de documents dont les mots clés contiennent m. La fonction `List.mem` peut-être utile ici.

   Réponse:

   ```
   let rec chercheUnMot m indx = match indx
   with  []   -> []
   | (t,l)::r -> if List.mem m l then t::(chercheUnMot m r)
   else chercheUnMot m r;;

   chercheUnMot "Egypte" i;;
   ```

2. la fonction `chercheTousLesMots` qui prend un index i et une liste de mots clés m, et qui renvoie la liste de tous les titres (et uniquement les titres) de documents qui contiennent tous les mots de m. Les fonctions `List.for_all` et `List.mem` peuvent être utiles ici.

   Réponse:

   ```
   let rec chercheTousLesMots lm indx = match indx
    with  []   -> []
      | (t,lt)::r -> if List.for_all (fun x -> List.mem x lt) lm
                          then t::(chercheTousLesMots lm r)
                 else chercheTousLesMots lm r;;

   chercheTousLesMots ["programmation";"informatique"] i;;
   ```

3. la fonction `pertinenceDe` qui prend une liste de mots l et une paire (t, lm) composé d'un titre et de sa liste de mots clés. La fonction retourne le nombre d'éléments de l qui sont dans lm, ce qui donne une mesure de la pertinence du document t par rapport à la liste de mots-clés l. Par exemple, `pertinenceDe ["Egypte"; "dieux"]` appelée sur `("Amon-Re",["Egypte"; "dieux"])` renvoie 2, et sur `("Voyager en Egypte",["Egypte"; "voyages"])` renvoie 1.
   Réponse:

```
let pertinenceDe lm (t,lt) =
 let rec compte lm =
 match lm
 with [] -> 0
   | m::r -> if List.mem m lt then 1+ (compte r)
             else compte r
 in compte lm;;

pertinenceDe ["Egypte"; "dieux"] ("Voyager en Egypte",["Egypte"; "voyages"]);;
```

4. Ecrivez la fonction `chercheParPertinence` qui prend une liste de mots-clés et un index et renvoie le document le plus pertinent de l'index, à savoir, celui dont la liste de mots clés contient le plus de mots recherchés.
   Réponse:

```
let lePlusPertinent lm indx =
 let pesageDocs = List.map (fun (t,lt) -> (t, pertinenceDe lm (t,lt))) indx in
 let rec maxDoc p =
 match p
 with [] -> failwith "lePlusPertinent"
  |  [(t,_) as x] -> x
  |  (t,n)::r -> let (tr,maxr) = maxDoc r in
                 if n>maxr then (t,n) else (tr,maxr)
 in fst(maxDoc pesageDocs);;

lePlusPertinent ["programmation";"informatique"; "Java"] i;;
```

□

Exercice 2 (4 points):

On souhaite modéliser le découpage en sections, sous-sections et sous-sous-sections d'un document. On utilise pour cela une structure d'arbre, où chaque noeud correspond à une section, identifiée par son titre (une chaîne), et auquel on associe la liste de ses sous-sections, elles mêmes pouvant avoir des sous-sections. Le type `section` permet de représenter ceci, où le constructeur `FinalSect` correspond à une section terminale dans sa hiérarchie. Un document est donc une liste de sections.

```
type section = FinalSect of string
         | Section of string * section list;;

type document = Document of section list;;

let d = Document [Section ("Amon-Re",
```

```
            [Section("Baa",
                [FinalSect "Canope"; FinalSect "Delphes"]);
             FinalSect "Euphrates"]);
          FinalSect "Fin"];;
```

1. Ecrire une fonction `afficheDoc` capable d'afficher la hiérarchie des sections et sousections d'un document. Ainsi, le document `d` de l'example, devra être affiché sous la forme :

```
# afficheDoc d;;
1 section Amon-Re
  1.1 subsection Baa
    1.1.1 subsection Canope
    1.1.2 subsection Delphes
  1.2 subsection Euphrates
2 section Fin
- : unit = ()
```

Réponse:

```
let afficheDoc (Document d) =
  let rec affiche d c mess cm =
  match d
  with [] -> ()
   |  (Section (title,ss))::r ->
          let num = cm^(string_of_int c) in
          let s = mess^"section␣" in
          let entete = num^s^title in
          print_string entete; print_newline();
          affiche ss 1 "␣sub" ("␣␣"^num^".");
          affiche r (c+1) mess cm
   |  (FinalSect title)::r ->
          let num = cm^(string_of_int c) in
          let s = mess^"section␣" in
          let entete = num^s^title in
          print_string entete; print_newline();
          affiche r (c+1) mess cm
  in affiche d 1 "␣" "";;
```

2. Ecrire une fonction `numerotationDe` qui prend un titre de section et un document, et renvoie la numérotation sous forme de chaîne de caractères, correspondant à ce titre dans le document. Exemple : `numerotationDe "Canope"` renvoie la chaîne `"1.1.1"`.

□

Exercice 3 (3 points):

Soit l'ensemble E défini par induction comme suit :

$$\begin{array}{rcl} B_E & = & \{0, 7\} \\ \Omega_B & = & \{\omega : (E \times E \times E) \to E\} \\ & & \text{où } \omega(i, j, k) = i + 7j + k. \end{array}$$

1. Démontrez que tous les éléments de E sont divisibles par 7. Autrement dit démontrez la propriété suivante : $\forall n \in E, \exists i \in \mathbb{N}, n = 7i$.
   Réponse: Par induction sur $n \in E$.
2. (plus dur) Démontrez que E contient tous les multiples positifs de 7. Autrement dit démontrez la propriété suivante : $\forall n \in \mathbb{N}, 7n \in E$.
   Réponse: Par récurrence *sur $n \in \mathbb{N}$!*

□

Exercice 4 (3 points):

Soit g la fonction suivante :

```
let rec somme x res =
  if x <= 0 then res
  else somme (x-1) (x+res);;
```

1. (2,5pt) Démontrez la propriété suivante : $\forall x, \forall res$ `somme` $x\ res = (\sum_{i=0}^{x} i) + res$
   Réponse: Par récurrence sur x.
   - *Base : $x = 0$, OK.*
   - *Réc : Supposons que pour un x quelconque $\forall res$,* `somme` $x\ res = (\sum_{i=0}^{x} i) + res$. *Montrons qu'alors $\forall res$,* `somme` $(x+1)\ res = (\sum_{i=0}^{x+1} i) + res$. *Soit res un entier positif quelconque, par définition* `somme` $(x+1)\ res =$ `somme` $x\ (x+1+res)$. *Or par hypothèse de récurrence* `somme` $x\ (x+1+res) = (\sum_{i=0}^{x} i) + (x+1+res) = (\sum_{i=0}^{x+1} i) + res$. *OK.*
2. (0,5pt) En déduire que $\forall$ `x`,`somme x 0` = $(\sum_{i=0}^{\mathtt{x}} i)$

□

Exercice 5 (4 points):

Soit f la fonction suivante :

```
let rec f (x,y) =
  if x<=0 or y<=0 then 0
  else if x < y then f (x,x)
  else 1 + f (y-1,y)
```

1. Démontrez que cette fonction termine sur $\mathbb{N} \times \mathbb{N}$.
   Réponse: On prend l'ordre $<_{\mathbb{N}}$ sur la somme des deux arguments.
2. Démontrez la propriété suivante : $\forall (x,y), f(x,y) \leq \sum_{i=1}^{x} i$. Vous procéderez par récurrence forte en traitant d'abord les cas $x = 0$ et $y = 0$.
   Réponse: Par récurrence forte sur la somme de x et y.
   - *Supposons que pour un $n \in \mathbb{N}$ quelconque, $\forall (x,y)$ t.q. $x + y < n, f(x,y) \leq \sum_{i=1}^{x} i$. Démontrons qu'alors $\forall (x,y)$ t.q. $x + y = n, f(x,y) \leq \sum_{i=1}^{x} i$.*
     Soient x et y tels que $x + y = n$. On distingue 4 cas :
     (a) $x = 0$ alors $f(x,y) = 0$ OK.
     (b) $y = 0$ alors $f(x,y) = 0$ OK.
     (c) $0 < x < y$, alors $f(x,y) = f(x,x)$, comme $x < y$ on a $x + x < n$ donc par hypothèse de récurrence, $f(x,x) \leq \sum_{i=1}^{x} i$. OK (puisque $f(x,y) = f(x,x)$).

(d) $x \geq y > 0$, *alors* $f(x,y) = 1+f(y-1,y)$. *Comme* $x \geq y$ *on a* $y-1+y < x+y$ *donc par hypothèse de récurrence,* $f(y-1,y) \leq \sum_{i=1}^{y-1} i$. *Donc* $1 + f(y-1,y) \leq (\sum_{i=1}^{y-1} i) + 1$. *Comme* $y > 0$, $(\sum_{i=1}^{y-1} i) + 1 \leq (\sum_{i=1}^{y-1} i) + y = \sum_{i=1}^{y} i$, *OK.*

□