Université Paris Diderot
M1 Informatique

Programmation Fonctionnelle Avancée
Année 2012-2013

TP n°4

Compréhensions de liste

Exercice 1

1. En utilisant la traduction donnée en cours, convertissez le programme suivant, écrit avec l'extension de syntaxe CamlP4, en syntaxe OCaml standard :

```
let rec range i j = if i > j then [] else i :: range (i+1) j;;
let sum_xy = [(x+y) | x <- range 1 10 ; y <- range 1 10];;
```

2. Ecrivez maintenant en utilisant des compréhensions de liste un code équivalent au suivant :

```
List.filter (fun x -> x > 5) (List.map (fun x -> 2*x) (range 1 10));;
```

Vérifiez sur quelques exemples que vous obtenez bien le même résultat.

Exercice 2

1. En utilisant l'extension CamlP4, écrivez des fonctions permettant de :
   - Récupérer tous les nombres pairs d'une liste.
   - Récupérer tous les fichiers d'un répertoire qui ont une taille plus grande qu'un entier donné.
   - Trouver les 4 chiffres A, B, C, D qui vérifient l'équation $ABCD*4 = DCBA$ où $ABCD$ et $DCBA$ sont deux nombres de quatre chiffres.
2. Reprenez les même questions mais en utilisant cette fois ci le module ListMonad.

Exercice 3 Nous avons trouvé les horaires de trois films qui nous intéressent, et les avons représentés dans la structure de données suivante, qui contient le nom du film, sa durée en minutes, ainsi que la liste des horaires, avec heure et minute de début du film :

```
let movies = ["Lincoln",149,[12,15;15,15;18,25;];
              "Django",164,[17,40;21,0];
              "Mobius",103,[12,50;15,0;17,20;19,50;22,10]];;
```

Nous voulons trouver les combinaisons de séances qui minimisent la distance entre les horaires de sortie. Pour cela :

- calculez les horaires de sortie de chaque seance

```
let mftimes = ...
val mftimes : ((string * (int * int)) * int) list list =
 [[(("Lincoln", (12, 15)), 884); (("Lincoln", (15, 15)), 1064);
   (("Lincoln", (18, 25)), 1254)];
  [(("Django", (17, 40)), 1224); (("Django", (21, 0)), 1424)];
  [(("Mobius", (12, 50)), 873); (("Mobius", (15, 0)), 1003);
   (("Mobius", (17, 20)), 1143); (("Mobius", (19, 50)), 1293);
   (("Mobius", (22, 10)), 1433)]]
```

- en utilisant la compréhension des listes, obtenez toutes les combinaisons de seances

```
let rec combinations l = ...
let combs = combinations mftimes;;
combs : ((string * (int * int)) * int) list list =
[[(("Lincoln", (12, 15)), 884); (("Django", (17, 40)), 1224);
  (("Mobius", (12, 50)), 873)];
 [(("Lincoln", (12, 15)), 884); (("Django", (17, 40)), 1224);
  (("Mobius", (15, 0)), 1003)]; ...
```

— calculez pour chaque combinaison l'attente maximale entre les sorties de séance

```
let maxwaitl l = ...
maxwaitl combs;;
- : (int * (string * (int * int)) list) list =
[(351, [("Lincoln", (12, 15)); ("Django", (17, 40)); ("Mobius", (12, 50))]);
 (340, [("Lincoln", (12, 15)); ("Django", (17, 40)); ("Mobius", (15, 0))]);
 (340, [("Lincoln", (12, 15)); ("Django", (17, 40)); ("Mobius", (17, 20))]);
 ...
```

— ordonnez les combinaisons en conséquence