Université Paris 7 - DEUG-MIAS Année 2002-2003

TP de *Logique et Circuits* n° 10

Induction sur les listes

Exercice 1

1. Ecrire une fonction `ajout_gauche` telle que `ajout_gauche x l` renvoie la liste obtenue en remplaçant chaque élément `e` de `l` par le couple `(x,e)`.

```
ajout_gauche 0 [1;2;3] = [(0,1);(0,2);(0,3)]
ajout_gauche 0 []      = []
```

2. A partir de cette fonction, écrire une fonction `produit` calculant le produit de deux listes.

```
produit ['a';'b'] [1;2;3] = [('a',1);('a',2);('a',3);('b',1);('b',2);('b',3)]
produit [] [1;2;3]        = []
```

Indication. Raisonner par induction sur la première liste.

Exercice 2 On souhaite implémenter en Caml le tri par fusion vu en TD.

1. Ecrire une fonction `separer`. Etant donné une liste `l`, `separer l` doit renvoyer un couple de listes `(l1,l2)`, chaque liste contenant la moitié des éléments de `l` (dans le cas des listes impaires, `l1` contiendra un élément de plus).

   Indication. On pourra stocker dans `l1` tous les éléments de `l` d'indice impair, et dans `l2` tous les éléments d'indice pair, en utilisant un match à 3 cas de la forme :

```
let rec separer l = match l with
  []       -> ...
| [a]      -> ...
| a::b::ll -> ...
;;
```

2. Ecrire une fonction `fusion` calculant la fusion de deux listes `l1` et `l2`, supposées déjà triées.

   Indication. Effectuer un match sur le couple `(l1,l2)`.

3. A l'aide des deux fonctions précédentes, écrire `tri_fusion`, calculant le tri par fusion de deux listes.

Exercice 3

1. Ecrire une fonction `prefixe` qui détermine si une liste est préfixe d'une autre.

```
prefixe [1;2] [1;2;3] = true
prefixe [1;2] [2;3]   = false
```

2. En déduire une fonction `occurrence` qui détermine si une liste a au moins une occurrence dans une autre.

```
occurrence [2;3] [1;2;3;4] = true
occurrence [3;2] [1;2;3;4] = false
occurrence [2;3] [1;2;4;3] = false
```