Programmation fonctionnelle 2021-2022

Feuille 8 : Plus sur les listes

Exercice 8.1 Les fonctions `sum`, `prod` et `op_prod` vues précédemment suivent le même schéma de récursion. On peut écrire deux fonctions pour décrire de tels schémas :

- Si `l` est la liste `[b1; ...; bn]`, `left_fold op a l` vaut `op (...(op (op a b1) b2)...) bn`.
- Si `l` est la liste `[a1; ...; an]`, `right_fold op l b` vaut `op a1 (op a2 (...(op an b)...))`.

Cet exercice demande de comprendre et programmer ces fonctions. Elles seront ensuite utilisées pour reprogrammer des fonctions déjà vues.

1. Pour comprendre ces schémas, calculer à la main
   - `left_fold op a l` et
   - `right_fold op l b`

   pour `op` la fonction `fun x y -> x + y`, `l` la liste `[1;2;3]`, et `a` et `b` valant `0`.
2. Écrire la fonction `right_fold`.
3. Écrire la fonction `left_fold`.

   Note : Ces fonctions existent dans la bibliothèque standard sous les noms `List.fold_right` et `List.fold_left`.

Exercice 8.2 En utilisant les fonctions `List.fold_left` et/ou `List.fold_right`, écrire ou réécrire les fonctions suivantes.

1. Une fonction `length l` qui calcule la longueur de la liste `l`.
2. Une fonction `reverse l` qui calcule la liste renversée de `l`.
3. Une fonction `maximum l` qui calcule le maximum de la liste d'entiers `l`.
4. Une fonction `filter p l`, où `p` est un prédicat s'appliquant aux éléments de `l`, qui calcule la liste des éléments de `l` qui satisfont le prédicat `p`.
5. Une fonction `remove_duplicates l` qui ne garde qu'une occurrence de chaque élément de `l`.
6. La fonction `append` qui concatène deux listes.
7. La fonction `map f l` écrite dans la feuille 1.