Feuille de TD n°2 : structure de liste

MPSI Lycée Clemenceau : option informatique

Mars 2025

Exercice 1 : Ecrire une fonction `make_list` de signature `int -> 'a -> 'a list` renvoyant une liste de n éléments identiques.

Exercice 2 : Ecrire une fonction qui retourne l'avant-dernier élément d'une liste, s'il existe.

Exercice 3 : Ecrire une fonction calculant la somme des éléments (entiers) d'une liste.

Exercice 4 : Ecrire une fonction calculant la liste de tous les préfixes d'une liste donnée.
Par exemple :

```
prefixes [1;2;3;4] ;;
- : int list list = [[1];[1;2];[1;2;3];[1;2;3;4]]
```

Exercice 5 : Ecrire une fonction `ocsmin : 'a list -> 'a * int = <fun>` qui renvoie le couple formé par l'élément minimum d'une liste et le nombre de fois où il apparaît.

Exercice 6 : Ecrire une fonction `flatten : 'a list list -> 'a list` aplatissant une liste de listes.
Par exemple, `flatten [[3; 4]; [5;7;9]; [0]; [6;8]]` doit renvoyer `[3; 4; 5; 7; 9; 0; 6; 8]`.

Exercice 7 : *Listes et ordre*

1) Ecrire un fonction `est_constante : 'a list -> bool`, qui indique si une liste est constante, c'est à dire tous ses membres sont égaux.

2) Ecrire une fonction `est_croissante` et une fonction `est_decroissante` qui indiquent si la liste transmise en argument est respectivement croissante ou décroissante au sens large.

3) Utiliser la question précédente pour écrire une fonction `est_monotone` qui spécifie si la liste est monotone au sens large.

4) Ecrire une fonction `est_monotone2` qui n'utilise pas la question 2 et qui spécifie si la liste est monotone au sens large en ne parcourant qu'au maximum qu'une fois la liste.

Exercice 8 : Ecrire une fonction `lfactors: int -> int list` renvoyant la liste des facteurs premiers d'un entier positif n (chaque facteur étant éventuellement répété autant que nécessaire).

Exercice 9 : On souhaite écrire une fonction `purge` qui, appliquée à une liste, retourne une liste dans laquelle les doublons ont été éliminés (on pourra utiliser la fonction prédéfinie `List.mem` qui détermine si un élément appartient ou pas à une liste, on fera ensuite une version qui redéfinit cette fonction).

1) Écrire une première version de purge dans laquelle seule la dernière occurrence de chaque doublon sera conservée. Par exemple, `purge [1; 2; 3; 1; 4; 3; 1]` renverra comme résultat `[2; 4; 3; 1]`.

2) Écrire une seconde version de purge dans laquelle seule la première occurrence de chaque doublon sera conservée. Cette fois, `purge [1; 2; 3; 1; 4; 3; 1]` renverra le résultat `[1; 2; 3; 4]`.

Exercice 10 : On peut définir une fonction qui teste des éléments devant vérifier une propriété si celle-ci est de signature `'a -> bool`.

1) Ecrire une fonction de signature `('a -> bool) -> int -> 'a list -> 'a`, qui donne le $n$ième élément d'une liste vérifiant une certaine propriété (s'il existe).

2) Ecrire une fonction de signature `('a -> bool) -> 'a list -> 'a` qui donne le dernier élément d'une liste vérifiant une certaine propriété (s'il existe).

Exercice 11 : On représente un ensemble par une liste, chaque élément de l'ensemble ne devant apparaître qu'une seule fois dans la liste, à un emplacement arbitraire.

1) Définir une fonction intersection qui calcule l'intersection de deux ensembles. Évaluer son coût en fonction des cardinaux de ces ensembles.

2) Définir de même l'union et la différence symétrique de deux ensembles.

3) Rédiger enfin une fonction égal qui détermine si deux ensembles sont égaux.

Exercice 12 : On représente un polynôme par une liste de type `( float * int ) list`.
Chaque élément de la liste représente un monôme, ceux-ci sont rangés dans la liste par ordre de degré croissant.
Ainsi le polynôme $\frac{3}{2}X^7 - 4X^3 + 5X^2$ est représenté par la liste `[ (5.,2) ; (4.,3) ; (1.5,7) ]`.

1) Quel est l'intérêt d'une telle représentation, par rapport aux tableaux vus lors du TD 1 ?

2) Définir les fonctions somme et produit pour cette représentation.