TP 5 : Algorithmique fonctionnelle
Polymorphisme et ordre supérieur

Hervé de Milleville - Stefan Bornhofen - Peio Loubière

18 février 2013

1 ... et bien d'autres !

Écrire une fonction d'insertion d'un élément trié dans une liste est aisé, vous allez d'ailleurs l'implémenter naïvement.
On souhaite, pour un gros programme, ne créer qu'une seule fonction d'insertion permettant de faire la même chose, de manière identique sur un liste de notes, une liste d'étudiants.
On souhaite, de même, insérer un étudiant dans une liste selon l'ordre alphabétique du nom de famille ou par moyenne décroissante. Écrire alors une unique fonction d'insertion générique permettant de réaliser toutes ces opérations (...et bien d'autres !) et écrire alors les relations d'ordre annexes.

2 Dichotomie, le retour !

Écrire une fonction qui, de manière dichotomique, trouve le zéro d'une fonction continue et monotone, dans un intervalle $[a, b]$ dans lequel elle s'annule.

3 À la dérive !

Écrire une fonction `derive`, curryfiée, qui détermine la dérivée d'une fonction sur un intervalle $[x, x + dx]$.
Écrire une fonction `f'`, application partielle de dérive pour dx = $1e^{-12}$.
Tester sur des fonctions mathématiques usuelles (sin, cos, carré, etc.) en des points remarquables.

4 Étudiants, diants diants !

- Écrire une fonction qui retourne, depuis une liste d'étudiants, un étudiant qui se nomme Serge.
- Écrire une fonction qui retourne, depuis une liste d'étudiants, un étudiant qui a eu 20 à un ds.

- Écrire une fonction qui retourne, depuis une liste d'étudiants, un étudiant qui a eu au moins 3 notes en dessous de la moyenne.
- Écrire une fonction qui retourne, depuis une liste d'étudiants, un étudiant dont la moyenne est supérieure à 17.
- Se demander si on ne pouvait pas écrire une fonction générique... le faire, si la réponse à la question précédente est "Bon sang, mais c'est bien sûr !".

5 Un nouveau tri !

Le but est d'écrire un algorithme de tri par extraction. Son principe est calqué sur celui du tri par sélection où l'on détermine le plus petit/grand élément d'une liste, à ceci près qu'on l'extrait de la liste et qu'on l'insère en début/fin d'une liste triée.
La répétition de cette opération trie la liste initiale selon la relation d'ordre définie.

- Écrire une fonction `selBorne` qui retourne l'élément borné (selon une relation d'ordre) dans une liste ainsi que la liste sans ce dernier, telle que :
  `val selBorne : 'a list -> ('a -> 'a -> bool) -> 'a * 'a list = <fun>`
- Écrire la fonction `triExtraction` qui trie une liste selon une relation d'ordre comme définie ci-dessus, telle que :
  `val triExtraction : 'a list -> ('a -> 'a -> bool) -> 'a list = <fun>`
- Trier une liste d'étudiants par nom, croissant, puis par moyenne, décroissante.