TP 4 : un peu de programmation impérative

Toutes vos fonctions seront d'abord écrites, après analyse, sur papier. Vous ne les testerez sur machine qu'après être bien sûrs d'avoir tout vérifié. Soyez honnêtes, comptabilisez le nombre de fois où le code compile et fonctionne du premier coup et le nombre d'essais inutiles... Le but est d'arriver à chaque fois à écrire le code juste du premier coup.

Consigne évidente : une fois le code écrit et compilé sans erreur, il est bien sûr évident que vous le testerez sur quelques exemples. Vous choisirez évidemment ces exemples pertinement, de manière à tester les cas limites en particulier.

I Échauffement sur les références

Écrire trois fonctions `get x` , `set x v` et `clone x` qui prennent chacune une référence `x` en paramètre et éventuellement un autre paramètre `v` , et qui respectivement rend la valeur référencée par `x` , fait que la référence `x` référence la valeur `v` , ou rend une autre référence référençant la même valeur que celle de `x` .

Vous déterminerez bien sûr le type de chacune des fonctions avant de vérifier sur votre machine.

II Échauffement sur les tableaux

II.1 Somme

Écrire une fonction `somme tableau` qui calcule la somme des éléments d'un tableau de flottants.

II.2 Inverse

Écrire une fonction `inverse tableau` qui rend un **nouveau** tableau dont les éléments sont dans l'ordre inverse.

Écrire une fonction `inverse_ep tableau` qui modifie **en place** le tableau reçu en paramètre de manière à inverser le sens des éléments. Cette fonction ne doit pas utiliser de tableau intermédiaire.

III Tri à bulles

Le tri à bulles d'un tableau consiste à comparer successivement toutes les paires de cases voisines, de la gauche vers la droite. Si les deux cases contiennent deux valeurs qui ne sont pas dans le bon ordre, on les échange.

On recommence alors cette manipulation n fois pour un tableau de taille n.

Exécuter à la main sur un petit exemple cette procédure et justifier le nom de tri à bulles.

À la fin de la première passe on constate que l'élément le plus grand est nécessairement en dernière position. À la deuxième passe le deuxième plus grand est en avant-dernière position, etc.

III.1 Première version

Écrire une fonction `tri_a_bulles tab` , brutale, qui effectue les n passes évoquées dans la description ci-dessus pour trier le tableau `tab` .

III.2 Deuxième version optimisée

Fréquemment le tableau est trié avant la dernière passe. Modifier la fonction précédente pour écrire une fonction `tri_a_bulles_optimise tab` de manière à arrêter le tri dès que possible.

III.3 Génération de gros tableaux aléatoires pour tester vos fonctions

Écrire une fonction `tableau_aleatoire n min max` qui rend un tableau de n entiers aléatoirement compris au sens large entre `min` et `max` (avec évidemment $min < max$). On pourra pour cela utiliser la fonction `Random.int n` qui renvoie un entier aléatoire entre 0 (inclus) et n (exclu).

Tester alors vos fonctions de tri sur de gros tableaux aléatoires (par exemple 1000 valeurs entre -100 et 100) ou encore plus gros.

IV Tri par dénombrement

On souhaite trier un tableau dont les éléments sont des entiers naturels strictement inférieurs à un entier k. La méthode proposée ici porte le nom de tri par dénombrement.

1. Écrire une fonction compte de type int -> int array -> int array qui prend en arguments un entier `k` et un tableau `t` d'entiers naturels strictement inférieurs à k, et qui renvoie un tableau de taille k dont l'élément d'indice i contient le nombre d'apparitions de l'entier i dans `t` .
2. En déduire une fonction `tri_denombrement` de type int -> int array -> unit telle que `tri_denombrement k t` trie **en place** le tableau `t` de n entiers naturels strictement inférieurs à k en temps $\mathcal{O}(n + k)$.
3. Si k n'était pas passé en paramètre, la recherche de sa valeur à partir de t dégraderait-elle signficativement la complexité de la fonction ?

V Si vous avez fini... un peu de diviser pour régner : tri fusion

Implémenter le tri fusion d'un tableau puis d'une liste en essayant seul, sans trop regarder votre cours...