

*Le dernier exercice, signalé avec \*, est le devoir maison à rendre pour cette semaine.*

Dans toute cette feuille d'exercices, si  $T$  est un tableau, alors les indices de ses cases sont numérotées à partir de 0.

**Exercice 1.** *Une opération sur les tableaux.*

On considère la fonction suivante qui prend en entrée un tableau d'entiers de taille  $n$  et le modifie.

```
1 static void fonc(int[] t){
2     for(int i = 0; i < t.length - 1; i++){
3         t[i+1] = t[i+1] + t[i];
4     }
```

1. Que donnera cette fonction appliquée aux tableaux  $\{1,1,1\}$  et  $\{0,1,2\}$ ?
2. Pour quels tableaux l'appel `fonc(t)` ne modifie-t-il pas  $t$ ?
3. Ecrivez une fonction `foncInverse(int [] t)` telle que  $t$  reste inchangé après l'appel `foncInverse(fonc(t))`.

**Exercice 2.** *Trouver l'erreur.*

1. Comment fait-on pour montrer qu'un algorithme est incorrect ? Qu'il est correct ?
2. La fonction suivante `Max2` prend en entrée un tableau d'entiers  $T$  de longueur  $n$  et est retournée supposément son deuxième plus grand élément. Montrez sur un exemple qu'elle n'est pas correcte et corrigez les erreurs.

```
1 static int Max2(int[] t){
2     int max, max2;
3     if(t[0]>t[1]){
4         max = t[0];
5         max2 = t[1];
6     }
7     else{
8         max = t[1];
9         max2 = t[0];
10    }
11    for(int i=2;i< n-2; i++){
12        if(t[i] > max){
13            max = t[i];
14        }
15        else{
16            if(t[i] > max2){
17                max2 = t[i] ;
18            }
19        }
20    }
21    return max2;
22 }
```

**Exercice 3. *Min et max.***

On souhaite trouver le plus petit et le plus grand élément d'un tableau d'entiers donné.

1. Écrivez l'algorithme qui retourne le plus grand élément d'un tableau. Combien fait-il de comparaisons pour un tableau à  $n$  éléments ?
2. Proposez un algorithme simple qui résout le problème en effectuant, pour un tableau de  $n$  éléments,  $2(n - 1)$  comparaisons. Exécutez à la main votre algorithme sur le tableau

$$T = \begin{array}{|c|c|c|c|c|c|} \hline 2 & 1 & -3 & 5 & 4 & 8 \\ \hline \end{array}$$

3. Un algorithme plus efficace est celui qui, itérativement :
  - compare entre eux deux éléments consécutifs du tableau.
  - compare le plus petit des deux au *min* courant, et met éventuellement à jour le *min* courant.
  - compare le plus grand des deux au *max* courant, et met éventuellement à jour le *max* courant.

Écrivez cet algorithme. Exécutez à la main votre algorithme sur le tableau  $T$  ci-dessus. Combien de comparaisons sont effectuées cette fois, en fonction de la longueur du tableau ?

4. Obtient-on quelque chose de meilleur si on applique la méthode du point précédent à des paquets de trois (au lieu de deux) éléments consécutifs du tableau ?

**Exercice 4. *Ordre lexicographique.***

Dans cet exercice, on considère qu'un mot (français) est un tableau de caractères, et que l'on peut connaître l'ordre (alphabétique) de deux caractères (en utilisant les opérateurs  $<$ ,  $<=$ ,  $=$ ,  $>=$  et  $>$ ). Par exemple, on a ' $a$ '  $<$  ' $b$ ', ' $b$ '  $=$  ' $b$ ', ' $c$ '  $>$  ' $b$ '. En revanche, on ne peut pas comparer directement deux mots.

1. Écrivez une fonction `inf` qui prend en argument deux mots et qui retourne -1 si le premier mot est avant le second dans l'ordre lexicographique (celui du dictionnaire), 0 si les deux mots sont les mêmes, et +1 sinon.
2. Écrivez une fonction `min` (utilisant `inf`) qui prend en argument un tableau de mots et qui retourne le plus petit mot pour l'ordre alphabétique.

**Exercice 5. *Médiane\**.**

La médiane d'un tableau  $T$  à  $n$  éléments distincts avec  $n$  impair, est l'élément  $x$  de  $T$  tel que exactement  $(n - 1)/2$  éléments de  $T$  sont strictement inférieurs à  $x$ .

1. Écrivez une fonction `countInf` qui prend en argument un tableau d'entiers  $T$  et un entier  $x$ , et qui retourne le nombre d'éléments de  $T$  qui sont strictement inférieurs à  $x$ .
2. Écrivez un algorithme (utilisant cette fonction) qui trouve la médiane d'un tableau  $T$  à  $n$  éléments distincts avec  $n$  impair.