T.P. 4

Les tableaux en C

L'exercice sur les tableaux qui suit, pour lequel la correction en pseudo-code est donnée, suppose d'écrire des fonctions d'entrées-sorties `LireTab` (lecture d'un tableau à l'entrée standard) et `EcrireTab` (écriture d'un tableau sur la sortie standard).

NB: Attention, dans la présentation de la méthode et dans les corrections en pseudo-langage, les tableaux sont indicés à partir de 1, alors qu'en C les tableaux sont indicés à partir de 0.

1 Exercice

Le but est d'écrire une procédure qui à partir d'un tableau `T` construit un tableau `Tlisse` qui est lissé (c'est-à-dire correspond à une moyenne mobile de `T`). Plusieurs variantes sont possibles suivant la méthode de calcul et le type de moyenne voulue (avec la position `p` de référence à gauche, au centre, à droite).

Nous allons supposer que nous travaillons avec position de référence à gauche. Nous noterons `k` la taille de la fenêtre permettant le lissage.

Exemple: Lissage avec k=3, référence à gauche de

T =	1	0	1	0	1	0	1	0	donne
Tlisse =	$\frac{2}{3}$	$\frac{1}{3}$	$\frac{2}{3}$	$\frac{1}{3}$	$\frac{2}{3}$	$\frac{1}{3}$	?	?	

Dans ce cas (référence à gauche) les deux (en général `k-1`) dernières valeurs ne sont pas définies. Nous les définirons en considérant :

$Tlisse[n-1] = (T[n-1] + T[n])/2$ et $Tlisse[n] = T[n]$ (ici 1/2 et 0).

1.1 Méthode triviale

$Tlisse[p] = (T[p] + T[p+1] + \ldots + T[p+k-1])/k$

```
procedure LisseTrivial(var T: Tab; var Tlisse: Tab; k: entier)
/* T passe par adresse pour economiser une recopie */
/* Tlisse va etre modifie donc passage par adresse obligatoire */
/* le tableau Tlisse contient a la fin le lissage par fenetre mobile */
/* de taille k de T. Ici la fenetre mobile est cadree a gauche */
/* lorsqu'il n'y a plus assez d'elements dans la fenetre on */
/* fait une moyenne sur la fenetre courante */
var p: entier
debut
pour p de 1 a N faire
     Tlisse[p] <-- 0
fpour
```

```
pour p de 1 a N-k+1 faire
     pour m de 0 a k-1 faire
          Tlisse[p] <-- Tlisse[p] + T[p+m]
     fpour
     Tlisse[p] <-- Tlisse[p] / k
fpour
/**/
pour p de N-k+2 a N faire
     m <-- 0
     tantque p+m <= N faire
          Tlisse[p] <-- Tlisse[p] + T{p+m]
          m <-- m+1
     ftantque
     Tlisse[p] <-- Tlisse[p] / m
fpour
fin
```

1.2 Méthode plus efficace

$Tlisse[1] = (T[1] + T[2] + \ldots + T[k])/k$
$Tlisse[p] = Tlisse[p-1] + (T[p+k-1] - T[p-1])/k$ pour $p > 1$ et $p \leq n-k+1$
$Tlisse[p] = (Tlisse[p-1] - (T[p-1]/(k-m+1))) * ((k-m+1)/(k-m))$
pour $p = n-k+1+m$ et m variant de 1 à $k-1$

```
procedure LisseEfficace(var T: Tab; var Tlisse: Tab; k: entier)
/* T passe par adresse pour economiser une recopie */
/* Tlisse va etre modifie donc passage par adresse obligatoire */
/* le tableau Tlisse contient a la fin le lissage par fenetre mobile */
/* de taille k de T. Ici la fenetre mobile est cadree a gauche */
/* lorsqu'il n'y a plus assez d'elements dans la fenetre on */
/* fait une moyenne sur la fenetre courante */
var p: entier
debut
pour p de 1 a N faire
     Tlisse[p] <-- 0
fpour
pour m de 0 a k-1 faire
     Tlisse[1] <-- Tlisse[1] + T[1+m]
fpour
Tlisse[1] <-- Tlisse[1] / k
/**/
pour p de 2 a N-k+1 faire
     Tlisse[p] <-- Tlisse[p-1] + ((T[p+k-1] - T[p-1]) / k)
fpour
/**/
pour m de 1 a k-1 faire
     p <-- N-k+1+m
     Tlisse[p] <-- (Tlisse[p-1] - (T{p-1] / (k-m+1))) * ((k-m+1) / (k-m))
fpour
```

```
fin
```

Question complémentaire :
On écrira une fonction `EcartAuLissage` qui calcule la moyenne des carrés des écarts entre un tableau et le tableau lissé correspondant, ainsi qu'une fonction `PointFixe` qui lisse répétitivement un tableau jusqu'à ce que l'écart au lissage soit suffisamment petit.

1.3 Recherche de mots

Soit T un tableau d'entiers. On dira que $(T[p] \dots T[p+k-1])$ est un mot de taille k qui a une occurence en position p.

Ecrire une fonction `RechercheMotk` qui recherche dans T la position p telle que $T[p] \dots T[p+k-1]$ soit maximal pour k fixé.

```
fonction RechercheMotk(var T: Tab; k: entier)
debut
OccurenceMax <-- 1
ValeurMax <-- 0
pour m de 0 a k-1 faire
     ValeurMax <-- ValeurMax + T[1+m]
fpour
ValeurCourante <-- ValeurMax
pour p de 2 a N-k+1 faire
     ValeurCourante <-- ValeurCourante + T[p+k-1] - T[p-1]
     si ValeurCourante > ValeurMax alors
          ValeurMax <-- ValeurCourante
          OccurenceMax <-- p
     fsi
fpour
retourner (OccurenceMax)
fin
```