Devoir 1
Algorithmique et Structures de données (GLIN301)

1 Définition

Etant donné un tableau de n chiffres, un **décalage circulaire** consiste à décaler la place des éléments d'un tableau en considérant que les deux extrémités du tableau (début et fin) communiquent. Un décalage est effectué avec un ***pas*** particulier, correspondant au nombre de cases vers la droite dont chaque nombre est décalé (tout nombre "débordant" la fin du tableau est réintroduit par le début du tableau. Le décalage se déroule toujours de la gauche vers la droite. On supposera aussi que **pas** $<$ **n**.

Exemple : depuis le tableau suivant

5	7	8	6	4	6	3	1

si on applique un décalage circulaire de $pas = 3$, on obtient le tableau suivant :

6	3	1	5	7	8	6	4

Question 1 ***(1 pt)*** *Donnez le tableau résultant d'un décalage circulaire de* $pas = 5$

Correction :

6	4	6	3	1	5	7	8

2 Un premier algorithme réalisant un décalage

On considère que le premier indice d'un tableau est ici 0, et que la dernière case d'un tableau de n cases est donc à l'indice $n-1$. Soit l'algorithme suivant :

```
DecalageCirculaire_1 (dr T : tableau de n nombres , d pas : entier)
i = 0
tant que i < n faire
   T [ (i + pas) modulo  n]  =  T [i modulo n]
   i = i + 1
fin tant que
```

Question 2 ***(1 pt)*** *Effectuez une trace de cet algorithme pour le tableau de l'exemple avec un pas = 3.*

0	2	5	6	8

Correction : l'algorithme effectue un traitement, mais n'utilisant pas de variable de swap temporaire, il efface certaines valeurs du tableau. Voici ce qu'on peut attendre des étudiants en terme de "trace" :

	i	$T[0]$	$T[1]$	$T[2]$	$T[3]$	$T[4]$
avant la boucle	0	0	2	5	6	8
après tour boucle 1	1	0	2	5	0	8
après tour boucle 2	2	0	2	5	0	2
après tour boucle 3	3	5	2	5	0	2
après tour boucle 4	4	5	0	5	0	2
après tour boucle 5	5	5	0	2	0	2

Question 3 ***(2 pt)*** *Prouvez l'arrêt de cet algorithme*

Correction : i est une expression qui croit d'une quantité (fixe) strictement positive à chaque tour de boucle, partant d'une valeur à distance finie de la valeur n, donc i dépasse donc cette valeur au bout d'un temps fini, ce qui termine l'exécution de cette boucle, et donc de l'algorithme.

Question 4 ***(2 pt)*** *Montrez pourquoi cet algorithme n'est pas correct pour résoudre le problème considéré.*

Correction : non utilisation d'une variable tmp de swap, donc écrasement de certaines valeurs

3 Une première correction

On propose de rédiger un algorithme correct pour le problème du décalage circulaire. Pour cela vous devez suivre strictement les indications données exercice après exercice (et ne pas proposer d'algorithme suivant un autre principe, pour l'instant ;-).

On décide d'utiliser un tableau auxiliaire (nommé R) dans lequel on recopie successivement les valeurs du premier tableau T.
A la fin de l'algorithme on veut que

1. $\forall i \ : \ 0 \leq i < pas \Rightarrow R[i] = T[i + n - pas]$
2. $\forall i \ : \ pas \leq i < n \Rightarrow R[i] = T[i - pas]$

Pour remplir ce tableau auxiliaire, on utilise deux boucles, exécutées l'une après l'autre :

- la première boucle recopie (dans le bon ordre) le bloc de toutes les valeurs qui sont à la fin du tableau T et qui se retrouvent au début de R (parce qu'elles sont décalées "trop" loin).
- la deuxième boucle recopie le bloc situé au début de T et qui se retrouve à la fin dans R.

Les deux boucles doivent respecter les invariants suivants :

- **Invariant de la première boucle** :
  à l'itération k : $\forall i \ : \ 0 \leq i < k \Rightarrow R[i] = T[i + n - pas]$

  Question 5 ***(2 pt)*** *Écrire cette première boucle "*`tant que k ...`*"*
  Indication *: demandez-vous quelle doit être la condition d'arrêt de cette première boucle (sur k) pour ne pas effectuer de dépassement de tableau dans T ?*
  Correction : condition d'arrêt : $k = pas$
  Algorithme :

```
k ← 0
tant que k < pas faire
   R [ k ]  =  T [ k + n - pas ]
   k ← k + 1
fin tant que
```

  Question 6 ***(2 pt)*** *Prouver l'invariant de boucle en procédant par récurrence.*
  Correction : l'invariant dit que toutes les valeurs de R sur l'intervalle

d'indices [0..k[ont été correctement recopiées à la fin du k-1ème tour de boucle, donc à l"entrée du kème tour. Au kème tour de boucle on recopie correctement R[k] et on incrémente k sans rien changer d'autre donc l'invariant est vrai à la sortie du kème tour de boucle.

– **Invariant de la deuxième boucle :**
à la k^{ieme} itération : $\forall i \ : \ i < k \Rightarrow R[i + pas] = T[i]$

Question 7 ***(3 pt)*** *Ecrire l'algorithme* `DecalageCirculaire_2` *contenant les deux boucles nécessaires (dont la 1ère écrite précédemment). Note : L'entête de l'algorithme sera rédigée de telle sorte que le tableau auxiliaire R puisse être communiqué à l'extérieur de l'algorithme rédigé ici, évitant ainsi une boucle supplémentaire pour recopier le contenu de R dans le tableau T.*

Correction :

```
DecCirc_2 (d T : tab de n nbres , r R : tab de n nombres , d pas : entier)
  k ← 0
  tant que k ≤ pas-1 faire
     R[k] ←  T[k+n-pas]
     k ← k + 1
  ftq
  k ← 0
  tant que k ≤ n-1-pas faire
    R[k+pas] ←  T[k]
     k ← k + 1
  ftq
 /* pas d'instruction renvoyer ici */
fin algo
```

Question 8 ***(2 pt)*** *Sur la base des invariants (on ne demande pas la preuve du deuxième invariant), justifiez la correction de l'algorithme.*

Correction :
• à la fin de la première boucle on a $\forall i \ : \ 0 \leq i < pas \Rightarrow R[i] = T[i+n-pas]$
• à la fin de la deuxième boucle on a $\forall i \ : \ 0 \leq i < n-pas \Rightarrow R[i+pas] = T[i]$
autrement dit pour la deuxième, $\forall i \ : \ pas \leq i < n \Rightarrow R[i] = T[i - pas]$

• ce qui donne ce qu'on demandait au début de la section (points 1 et 2), traduisant que le tableau R est bien un décalage circulaire du tableau T avec le pas demandé.

4 Raccourcis dans la solution précédente

On cherche maintenant une solution n'utilisant qu'**une seule boucle** (et non deux comme dans la question précédente). Indication : on utilisera les modulos pour remplacer les deux boucles par une seule.

Question 9 ***(3 pt)*** *Ecrivez l'algorithme* `DecalageCirculaire_3` *correspondant.*

Correction :

```
DecalageCirculaire_3 (d T : tableau de n nombres ) : entier
On supposera n>1
i ← 0
tant que i < n faire
   R[ (i+pas) mod n ] ← T[i]
   i ← i + 1
ftq
renvoyer d
```

Question 10 ***(2 pt)*** *Donnez (sans le prouver) l'invariant de la boucle de l'algorithme précédent*

Correction :

$$\forall i,\ 0 \leq i < k \quad R[(i + pas)\ mod\ n] = T[i]$$

5 Détection de décalage (bonus)

On suppose maintenant que l'on récupère un tableau qui était initialement **trié par ordre croissant** puis qui a subi un décalage d'un certain pas k inconnu, que l'on veut deviner. On suppose de plus que le tableau ne contient que des valeurs différentes.

Question 11 ***(2 pt)*** *Ecrivez un algorithme qui prend en entrée un tableau ayant subi un décalage et qui renvoie le pas k qui a été utilisé par ce décalage. On pourra s'inspirer de l'algorithme de recherche par dichotomie d'un nombre dans un tableau trié, vu en cours, et modifiant progressivement la portion $[g, d]$ examinée dans le tableau. Initialement on considère tout le tableau ($g = 0$,$d = n - 1$), puis chaque tour de boucle réduit la portion examinée (en modifiant soit g soit d). Les invariants de boucle seront les suivants :*

- $g < d$
- $T[g] \geq T[d]$

Correction : On veut finir dans la situation ou g pointe sur la plus grande valeur du tableau, et d sur la plus petite, située donc juste après g. On écrit la condition d'arrêt en conséquence. Notons que lorsque l'on accepte de faire un tour de boucle (c-a-d la condition de continuation est vérifiée), g et d sont séparés d'au moins une case ce qui garantit que la valeur mil calculée est strictement à droite de g, de plus elle est strictement à gauche de d. Ceci garantit qu'à ce tour de boucle, soit g progresse vers la droite, soit d progresse vers la gauche, montrant donc qu'à un moment donné la condition de continuation sera fausse, donc ceci prouve l'arrêt de l'algorithme.

```
DevineDecalageCirculaire (d T : tableau de n nombres ) : entier
On supposera n>1
g ← 0
d ← n-1
tant que g < d-1 faire
   mil← (g+d) div 2  /* division entière */
   si T[mil] > T[d]
       alors g← mil
       sinon d← mil
   fsi
ftq
renvoyer d
```

Question 12 ***(1 pt)*** *Donnez la complexité de votre algorithme en indiquant une ou deux phrases justifiant cette complexité.*

Correction : Complexité en $O(\log n)$ car la portion du tableau à examiner diminue de moitié à chaque étape.