IUT de Montpellier **Partiel**
Algorithmique avancée **Durée : 1h**

Pour tous les exercices il est interdit d'utiliser des boucles ou de créer vos propres méthodes auxiliaires. Par contre, vous pouvez utiliser les questions précédentes pour répondre à la question courante. Pour les tiers temps, les questions marquées par $\star$ ne sont pas à faire.

Les exercices sont classés par difficulté croissante (ou du moins qui me semble croissante).

Exercice 1. Echauffement.

Question 1.1.

Ecrire (sans utiliser d'operateur de modulo (tel %) ou de division (tel /)) le code de la méthode suivante

```
public static int modulo3(int x){
//prérequis :
// 0 <= x
//action :
// retourne x modulo 3, c'est à dire le reste dans la division
euclidienne de x par 3
```

Par exemple :

- `modulo3(5)` retourne 2
- `modulo3(9)` retourne 0

Exercice 2. Comparaison lexicographique.

On supposera que l'on peut comparer 2 caractères avec l'opérateur "<". Par exemple :

```
char c1 = 'b';
char c2 = 'x';
System.out.println ("test 1 :" + (c1 < c2)); //affiche "test 1 :  true"
System.out.println ("test 2 :" + (c1 < c1)); //affiche "test 2 :  false"
```

Question 2.1.

Ecrire le code de la méthode suivante

```
public static int compareMotsAux(char[] t1, char[] t2, int i){
//prérequis :
// t1.length==t2.length
// 0 <= i <= t1.length
//action :
// retourne
//-1 ssi le mot du sous tableau t1[i..(t.length-1)] est "strictement
plus petit" (c'est à dire placé avant dans le dictionnaire) que le
mot du sous tableau t2[i..(t.length-1)],
// 1 si il est "strictement plus grand",
// et 0 si ils sont égaux
```

Par exemple :

- `compareMotsAux({'b','i','l','l','e'},{'b','i','l','z','a'},0)` retourne -1 car le sous mot "bille" est avant "bilza" dans le dictionnaire
- `compareMotsAux({'a','d','r','e','t','a','g'},{'c','x','r','e','s','a','g'},2)` retourne 1 car le sous mot "retag" est après "resag" dans le dictionnaire

- `compareMotsAux({'a','d','r','e','t','a','g'},{'c','x','r','e','s','a','g'},5)` retourne 0 car les deux sous mots sont égaux (ils valent "ag")

Question 2.2.
En déduire le code de la méthode suivante

```
public static int compareMots(char[] t1, char[] t2){
//prérequis :
// t1.length==t2.length
//action :
// retourne -1 ssi le mot de t1 est "strictement plus petit"
// (c'est à dire placé avant dans le dictionnaire) que le mot de t2,
// 1 si il est "strictement plus grand", et 0 si ils sont égaux
```

Exercice 3. Séquence de zéros Le but final de cet exercice est de calculer le plus grand nombre de 0 consécutifs dans un tableau d'entiers.

Question 3.1.
Ecrire la méthode suivante :

```
public static int aux(int[] t, int i)
//prérequis :  0 <= i <= t.length
//action :  retourne un entier >=0 correspondant au nombre
de 0 consécutifs à partir de la position i (dans le tableau
t[i..t.length-1]).  Autrement dit, si aux(t,i) retourne x, alors
t[i]=t[i+1]=..=t[i+x-1]=0, et t[i+x]!=0
```

Par exemple, avec t = {0,0,0,0,0,1,2,0,0,0,0,3,6,0,0,9}

- aux(t,3) retourne 2
- aux(t,4) retourne 1
- aux(t,5) retourne 0
- aux(t,6) retourne 0
- aux(t,7) retourne 4

Question 3.2.
($\star$) En utilisant la question précédente, écrire la méthode suivante :

```
public static int nbConsec(int[] t, int i)
//prérequis :  0 <= i <= t.length
//action :  retourne le plus grand nombre de 0 consécutifs dans
le sous tableau t[i..t.length-1] (attention la séquence de 0
correspondante n'est pas obligée de commencer à la case i)
```

Par exemple, avec t = {0,0,0,0,0,1,2,0,0,0,0,3,6,0,0,9}

- nbConsec(t,3) retourne 4 car le plus grand nombre de 0 consécutifs dans le sous tableau t[3..15] est de 4
- nbConsec(t,10) retourne 2 car le plus grand nombre de 0 consécutifs dans le sous tableau t[10..15] est de 2

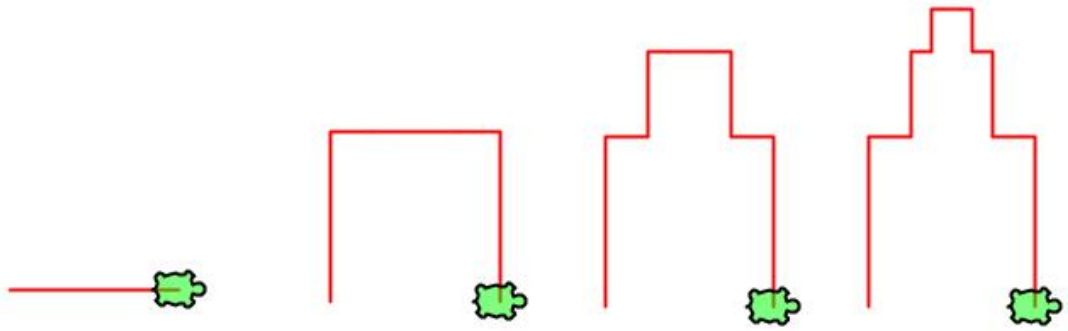

Figure 1: Exemples de tour gigogne de taille l et d'ordre 0, 1, 2 et 3 (de gauche à droite).

Exercice 4. Exercice de fractale : Tour Gigogne On appelle *tour-gigogne* de largeur l et d'ordre n une tour à n étages dont le premier étage est de hauteur l et de largeur l, la hauteur et la largeur étant divisée par 2 à chaque passage à l'étage supérieur (voir Figure 1).

On rappelle que la tortue bob se déplace (avec un crayon sous le ventre) à l'aide des méthodes suivantes :

- `bob.forward(x)` pour avancer de x pas,
- `bob.left(a)` et `bob.right(a)` pour tourner de a degrés.

Dans cet exercice on n'a pas besoin de lever le crayon.

Question 4.1.
Ecrire la méthode :

```
void tourGigogne (double l, int n)
// pré-requis : l ≥ 0 et n ≥ 0, le crayon est posé sur le sol et bob
regarde vers la droite
// action : fait tracer à bob le contour de la tour-gigogne de
largeur l et d'ordre n en position verticale vers le haut (voir
figure). A la fin, le crayon est posé sur le sol et bob regarde vers
la droite
```

Question 4.2.
($\star$)Ecrire la méthode :

```
double hauteurTG (double l, int n)
// pré-requis : l ≥ 0 et n ≥ 0
// résultat : retourne la hauteur de la tour-gigogne de largeur l et
d'ordre n
```

Question 4.3.
($\star$)Ecrire la méthode : `double longueurTG (double l, int n)`

```
// pré-requis : l ≥ 0 et n ≥ 0
// résultat : retourne la longueur du contour de la tour-gigogne de
largeur l et d'ordre n
```