IUT de Montpellier : AS 2017 **Partiel**
Algorithmique avancée **Durée : 2h**

Pour tous les exercices il est interdit d'utiliser des boucles ou de créer vos propres méthodes auxiliaires.

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

Exercice 1. Tableau alterné. On dit qu'un tableau de boolen est alterné ssi il contient alternativement vrai et faux, ou autrement dit ssi il ne contient pas deux "vrais" consécutifs ou deux "faux" consécutifs. Par exemple, $[t, f, t, f]$ est alterné, $[f, t]$ est alterné, et $[f, t, t, f, t]$ n'est pas alterné.

Question 1.1.
Ecrire la méthode suivante :

```
public static boolean alterneAux(boolean []t, int i)
//prérequis : 0 <= i < t.length
//action : retourne vrai ssi le sous tableau t[i..t.length-1] est
alterné
```

Par exemple, avec t = [f,t,t,f,t],

- alterneAux(t,3) retourne vrai
- alterneAux(t,2) retourne vrai
- alterneAux(t,1) retourne faux
- alterneAux(t,0) retourne faux

Question 1.2.
En déduire le code (non récursif) de la méthode suivante :

```
public static boolean alterne(int []t)
//prérequis : t non vide
//action : retourne vrai ssi t est alterné
```

Exercice 2. Retournement d'une chaîne. Dans cet exercice on s'autorise l'utilisation des méthodes (de la classe String) et de l'opérateur suivants :

- `int length()` : renvoie le nombre de caractère de la chaîne
- `String subString(int i, int j)` : renvoie la sous chaine comprise entre le i^{me} et le j^{me} caractère (avec $0 \leq i \leq j \leq length() - 1$)
- l'opérateur +, qui effectue la concaténation entre deux chaînes

Par exemple, si $s =$ "$abcd$", alors

```
int x = s.length(); // x vaut 4
String s2 = s.subString(1,1); // s2 vaut "b"
String s3 = s.subString(2,s.length()-1); // s3 vaut "cd"
String s4 = s3 + s2; //s4 vaut cdb
```

Question 2.1.
En utilisant les éléments ci dessus, écrire la méthode suivante :

```
public static String retourne(String s)
//prérequis : s != null (mais s = "" est autorise)
//action : renvoie une chaîne correspondant à s retournée
```

Par exemple, retourne("abcd") renvoie "dcba".

Exercice 3. 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).

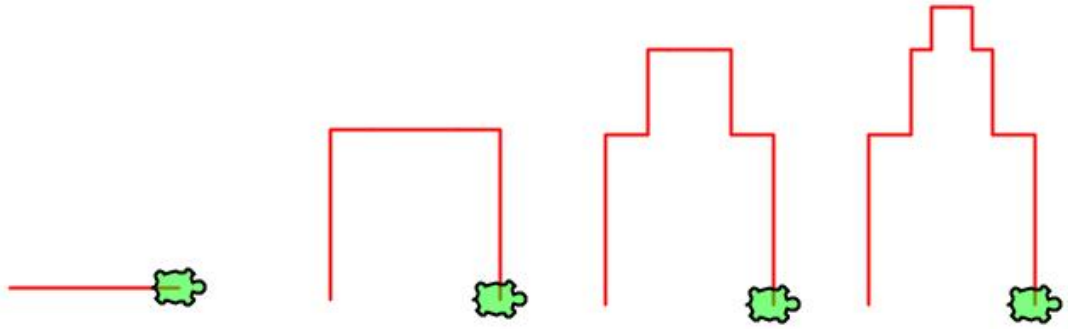

FIGURE 1 – Exemples de tour gigogne de taille l et d'ordre 0, 1, 2 et 3 (de gauche à droite).

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 a pas besoin de lever le crayon

Question 3.1.
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 3.2.
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 3.3.
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
```

Exercice 4. Tri d'un tableau à deux valeurs

Question 4.1.

Ecrire le code de la méthode suivante :

```
public static int triAux(int[] t, int i)
//prérequis :
// t ne contient que des 0 et des 1
// 0 <= i <= t.length
//action :
// modifie t[i..t.length-1] pour placer d'abord tous les 0 puis tous
les 1,
// et retourne l'indice du premier 1 dans le nouveau sous tableau
t[i..length-1] (et retourne t.length si il n'y a pas de 1)
```

Par exemple :

- triAux([1,0,1,1,1,0,0],2) transforme t en [1,0,0,0,1,1,1] et retourne 4
- triAux([1,0,1,1,1,0,0],5) transforme t en [1,0,1,1,1,0,0] et retourne 7