**IUT de Montpellier** **TD2 : Diviser pour régner**
**M3103 Algorithmique avancée**

**Exercice 1. Exponentiation rapide**

**Question 1.1.**
Rappeler l'algorithme récursif naïf `int puiss(int x, int n)` qui pour tout $n \geq 0$ et tout entier $x$ calcule $x^n$.

**Question 1.2.**
Nous allons maintenant appliquer le principe de diviser pour régner pour calculer plus rapidement $x^n$ : cela s'appelle ici l'exponentiation rapide. L'idée est de constater que $x^n = x^{\frac{n}{2}} x^{\frac{n}{2}}$. En vous inspirant de cette remarque, écrire une algorithme `int puissRapide(int x, int n)` qui pour tout $n \geq 0$ et tout entier $x$ calcule $x^n$.

**Exercice 2. Quicksort** On considère un tableau d'entier $t$ (que l'on souhaite trier par ordre croissant). Soit $p = t[t.length-1]$ que nous appellerons le pivot. L'idée du quicksort est la suivante. On va d'abord séparer $t$ en deux selon la valeur $p$ : pour un certain $c$ on placera les valeurs plus petites ou égales à $p$ dans les cases 0 à $c$ (avec la valeur $p$ dans la case $c$), et les valeurs strictement plus grandes que $p$ dans les cases $c+1$ à $t.length-1$. On obtient donc deux sous tableaux plus petits ($t[0..(c-1)]$ et $t[(c+1)..(t.length-1)]$), et il n'y a plus qu'à les trier récursivement!

**Question 2.1.**
Ecrire un algorithme (récursif ou non, au choix) `int pivot(int []t, int i, int j)` qui pour tout $t$ non vide, et tout $i$ et $j$ tels que $0 \leq i \leq j < t.length$, retourne une valeur $c$ et réorganise les éléments de $t[i..j]$ en mettant ($p$ dénote $t[j]$)

- les valeurs plus petites ou égales à $p$ dans les cases $i$ à $c$, avec la valeur $p$ dans la case $c$
- et les valeurs strictement plus grandes que $p$ dans les cases $c+1$ à $j$.

**Question 2.2.**
Ecrire un algorithme récursif `void quickSortAux(int []t,int i, int j)` qui pour tout tableau $t$ non vide, pour tout $i$ et $j$ tels que $0 \leq i$ et $j < t.length$ (on peut donc avoir $i \geq j$) trie le sous tableau $t[i..j]$ selon le principe du quicksort.

**Question 2.3.**
En déduire un algorithme `void quickSort(int []t)` qui trie $t$ selon le principe du quicksort.

# Exercices bonus

**Exercice 3. Tri shadok** Pour trier (par ordre croissant) un tableau d'entier $t$, les shadoks trient d'abord les deux premiers tiers du tableau, puis les deux derniers tiers, puis les deux premiers tiers à nouveau. Ecrire un algorithme récursif `void triShadok(int []t,int i, int j)` qui pour tout tableaut $t$, pour tout $i$ et $j$ tels que $0 \leq i < j < t.length$ (ou $i \leq j$ si vous préférez!) trie le sous tableau $t[i..j]$ selon le principe du tri shadok.