Exercice 1 (Compteur de voyelles)
Écrire une fonction `maxVowel` *qui prend en paramètre une chaîne de caractères* `s` *et renvoie la quantité de la voyelle (sans accents) qui apparaît le plus souvent. Par exemple,* `maxVowel("java")` *doit renvoyer* `2` *et* `maxVowel("bonjour tout le monde")` *doit renvoyer* `4`.

Exercice 2 (Scrabble)
Le score d'une lettre est sa position dans l'alphabet. Ainsi, le score de la lettre a est `1`*, celui de b est* `2`*, etc.... Le score d'un mot est le score de ses lettres. Écrire une fonction* `scrabble` *qui prend en paramètre une chaîne de caractères* `mot` *et qui renvoie son score. Par exemple,* `scrabble("abc")` *doit renvoyer* `6` *et* `scrabble("java")` *doit renvoyer* `34`.

Exercice 3 (+1 en binaire) *Écrire une fonction* `nextBin` *qui prend en entrée une chaîne de caractères représentant un nombre écrit en binaire, et renvoie une chaîne représentant le nombre suivant. Par exemple,* `nextBin("0010")` *doit renvoyer* `"0011"`*, et* `nextBin("0011")` *doit renvoyer* `"0100"`*. Attention au cas où il n'y a que des* `'1'`*, la longueur de la chaîne augmente :* `nextBin("1111")` *doit renvoyer* `"10000"`.

Exercice 4 (Changement de base) *Écrire une fonction* `versBase10` *qui prend en argument une chaîne de caractères* `s` *contenant un nombre écrit en base 2, et qui retourne l'entier correspondant. Par exemple,* `versBase10("1010")` *devra retourner 10, et* `versBase10("111")` *devra retourner 7.*

Exercice 5 (Répétitions)
Écrire une fonction `estUneRepetitionDe` *qui prend en paramètre deux chaînes de caractères* `s` *et* `t` *et renvoie* `true` *si* `s` *est une répétition de la chaîne de caractère* `t`*. Par exemple, l'appel* `estUneRepetitionDe("rougerougerouge","rouge")` *doit renvoyer* `true`*, alors que l'appel* `estUneRepetitionDe("vertver","vert")` *doit renvoyer* `false`.

Exercice 6 (Redondance) *Écrire une fonction* `repetitionDistanceK` *qui prend en entrée un entier* `K`*, et un tableau de chaînes de caractères* `t`*. Le tableau* `t` *représente une phrase découpée en mots. La fonction renvoie* `true` *si* `t` *contient deux fois le même mot, avec moins de* `K` *mots d'écart entre les deux. Par exemple, si* `t` = {`"Trop", "de", "Java", "tue", "le", "Java"`}*, alors un appel à la fonction* `repetitionDistanceK(3, t)` *doit renvoyer* `true`*, mais un appel à la fonction* `repetitionDistanceK(2, t)` *doit renvoyer* `false`.

Exercice 7 (Somme de sous-tableau) *Écrire une fonction* `reachSum` *qui prend en argument un tableau d'entiers* `t` *et un entier* `n`*. Le but de l'exercice est de déterminer s'il existe une "tranche" du tableau* `t` *dont la somme des éléments est égale à* `n`*. Plus précisément, on cherche à trouver deux indices* `i` $\leq$ `j` *tels que* `t[i] +...+ t[j] == n`*. Si c'est le cas, la fonction doit renvoyer* `true`*, sinon, on renvoie* `false`*. Par exemple, si* `t` = {`1, 5, -4, 6, 2, 12`}*, l'appel* `reachSum(t, 4)` *doit renvoyer* `true` *car* `-4+6+2 == 4`*, alors qu'un appel à* `reachSum(t, 3)` *renvoie* `false` *car il n'est pas possible d'obtenir* `3` *en faisant une somme d'éléments successifs de* `t`.

Exercice 8 (Sablier) *Écrire une procédure* `sablier` *qui prend en entrée un entier* `n`*, et affiche un sablier de hauteur* `n` *composé du symbole* `*`*. Par exemple, pour* `n = 7`*, on affichera :*

```
*******
 *****
  ***
   *
  ***
 *****
*******
```

Exercice 9 (Pivot) *Écrire une fonction* `plusPetitsQue` *qui prend en entrée un entier* `n`*, et un tableau d'entiers* `t`*, et qui renvoie le tableau constitué de tous les éléments de* `t` *qui sont strictement plus petits que* `n`*. Par exemple, si* `t = {0,8,1,6,5,3,9,0}`*, alors un appel à la fonction* `plusPetitsQue(5, t)` *doit renvoyer le tableau* `{0,1,3,0}`

Exercice 10 (Aplatir) *Écrire une fonction* `flatten` *qui prend en argument un tableau de tableaux d'entiers* `t`*, et qui renvoie un tableau d'entiers contenant tous les éléments de* `t`*. Attention, les sous-tableaux de* `t` *ne sont pas forcément tous de même longueur.*
Par exemple, si `t = {{1,2}, {3}, {4,5,6,7}, {8,9}}`*, alors un appel à la fonction* `flatten(t)` *doit renvoyer* `{1,2,3,4,5,6,7,8,9}`*.*

Exercice 11 (Matrice d'entiers) *Le but de cet exercice est d'écrire une fonction* `makeMatrix` *qui prend en argument un tableau de tableaux d'entiers* `t`*. Les sous-tableaux contenus dans* `t` *ne sont pas forcément tous de la même longueur : on souhaite les compléter en rajoutant des zéros, de manière à obtenir une matrice rectangulaire. Par exemple, si* `t = {{1, 2}, {3, 4, 5}, {6}}`*, alors un appel à* `makeMatrix(t)` *doit renvoyer le tableau* `{{1, 2, 0}, {3, 4, 5}, {6, 0, 0}}`*.*
Indication: *on pourra d'abord écrire une fonction auxiliaire qui calcule la largeur de la matrice finale.*

Exercice 12 (Complément) *Écrire une fonction* `complement` *qui prend en entrée un tableau d'entiers positifs* `t` *et un entier* `n`*, et qui renvoie un tableau d'entiers contenant tous les entiers positifs plus petits que* `n` *et qui ne sont pas contenus dans* `t`*. Par exemple, si* `t = {1, 5, 3}` *et* `n = 8`*, alors un appel à la fonction* `complement(t, n)` *doit renvoyer le tableau* `{2, 4, 6, 7}`*.*

Exercice 13 (Différence symétrique) *Écrire une fonction* `differenceSymetrique` *qui prend en entrée deux tableaux d'entiers* `s` *et* `t`*, et qui renvoie un nouveau tableau d'entiers comprenant tous les entiers contenus dans* `s` *ou dans* `t`*, mais pas dans les deux. Par exemple, si* `s = {1, 5, 4, 7}` *et* `t = {5, 7, 3, 2, 3, 7}`*, alors un appel à la fonction* `differenceSymetrique(s, t)` *doit renvoyer le tableau* `{1, 4, 3, 2, 3}` *à ordre près des éléments.*
Indice: *on pourra d'abord écrire une fonction qui calcule la taille du tableau final.*

Exercice 14 (Carré magique) *On dit qu'une matrice est un carré magique si la somme des éléments d'une de ses lignes ou de l'une de ses colonnes donne toujours la même valeur (on ne s'intéresse pas aux sommes des diagonales). Écrire une fonction* `estUnCarreMagique` *qui prend en entrée une matrice carrée* `t`*, et qui renvoie* `true` *s'il s'agit d'un carré magique et* `false` *sinon.*
Par exemple, si `t = {{1, 4, 5}, {3, 2, 5}, {6, 4, 0}}`*, alors l'appel* `estUnCarreMagique(t)` *doit renvoyer* `true`*.*

Exercice 15 (Matrice de permutation) *On dit qu'une matrice d'entiers est une* matrice de permutation *si : (a) elle ne contient que des 0 et des 1, et (b) sur chaque ligne et sur chaque colonne, la valeur 1 apparaît exactement une fois. Écrire une fonction* `estUneMatriceDePermutation` *qui prend en entrée une matrice carrée* `t`*, et qui renvoie* `true` *s'il s'agit d'une matrice de permutation et* `false` *sinon. Par exemple, si* `t = {{1, 0, 0}, {0, 0, 1}, {0, 1, 0}}`*, alors un appel à la fonction* `estUneMatriceDePermutation(t)` *doit renvoyer* `true`*.*

Exercice 16 (Permutation) *Écrire une procédure* `permutation` *qui prend en entrée deux tableaux d'entiers* `t` *et* `p` *de même longueur. On suppose que le tableau* `p` *contient une permutation de tous les entiers de* `0` *à* `t.length - 1`*. Un appel à la procédure* `permutation(t, p)` *doit mélanger les éléments du tableau* `t` *comme indiqué par la permutation* `p` *: la valeur qui était stockée initialement dans la case numéro* `i` *de* `t`*, doit se retrouver dans la case numéro* `p[i]` *à la fin de l'exécution du programme.*
Par exemple, si `t = {4, 8, 15, 16, 23, 42}` *et* `p = {3, 5, 1, 0, 2, 4}`*, alors après avoir exécuté la procédure* `permutation(t, p)`*, on doit avoir* `t = {16, 15, 23, 4, 42, 8}`*.*

Exercice 17 (Insertion) *Écrire une fonction* `insert` *qui prend en entrée un tableau d'entiers* `t`*, et un entier* `n`*. On suppose que le tableau* `t` *est trié par ordre croissant. Une appel à la fonction* `insert(t, n)` *doit renvoyer un nouveau tableau d'entiers, qui contient tous les éléments de* `t`*, plus l'élément* `n` *qui doit être inséré au bon endroit. Par exemple, si* `t = {0, 3, 6, 7, 11, 23}`*, l'appel* `insert(t, 9)` *doit renvoyer* `{0, 3, 6, 7, 9, 11, 23}`*.*

Exercice 18 (Transposition) *Écrire une fonction* `transpose` *qui prend en argument un tableau de tableaux d'entiers* `t`*. On suppose que tous les sous-tableaux contenus dans* `t` *sont de même longueur, de sorte que* `t` *peut être vu comme une grille rectangulaire d'entiers. Un appel à la fonction* `transpose` *doit renvoyer un nouveau tableau de tableaux qui contient les mêmes éléments que* `t`*, mais où on a échangé le rôle des lignes et des colonnes.*
Par exemple, si `t = {{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}}`*, alors un appel à la fonction* `transpose(t)` *doit renvoyer* `{{1, 5, 9}, {2, 6, 10}, {3, 7, 11}, {4, 8, 12}}`*.*

Exercice 19 (Somme par colonnes) *Écrire une fonction* `columnSum` *qui prend en argument un tableau de tableaux d'entiers* `t`*. On suppose que tous les sous-tableaux contenus dans* `t` *sont de même longueur, de sorte que* `t` *peut être vu comme une grille rectangulaire d'entiers. Un appel à la fonction* `columnSum` *doit renvoyer un tableau d'entiers dont la case numéro* `i` *contient la somme des éléments sur la* `i`*-ème colonne de* `t`*.*
Par exemple, si `t = {{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}}`*, alors un appel à la fonction* `columnSum(t)` *soit renvoyer* `{15, 18, 21, 24}`*.*

Exercice 20 (Décalage) *Écrire une procédure* `decalage` *qui prend en entrée un tableau d'entiers* `t` *et un entier* `k`*. Un appel à la procédure* `decalage(t, k)` *doit décaler tous les éléments du tableau* `t` *de* `k` *cases vers la droite. Les éléments qui dépassent la taille du tableau* `t` *doivent être insérés au début, de façon circulaire.*
Par exemple, si on a initialement `t = {4, 8, 15, 16, 23, 42}`*, alors après avoir exécuté la procédure* `decalage(t, 2)`*, on doit avoir* `t = {23, 42, 4, 8, 15, 16}`*.*
