❑ Exercice : *type A*

1. Rappeler le principe de l'implémentation d'un tableau associatif (ou dictionnaire) à l'aide d'une table de hachage.
2. Donner la définition des *collisions* et donner brièvement (au moins) un moyen de résoudre le problème qu'elles posent.

   Pour une chaine de caractères $s = c_0 \dots c_{n-1}$, on considère la fonction de hachage :

   $$h(s) = \sum_{i=0}^{n-1} 31^i \times c_i$$

3. Calculer le hash de la chaine `"AB"`.
4. Montrer qu'il existe deux chaines de caractères de longueur 2, formées de lettres minuscules (code 97 à 122) ou majuscules (code 65 à 90) et produisant la même valeur pour h.
5. En déduire une façon de construire un nombre arbitraire de chaînes de caractères de longueurs quelconques ayant la même valeur pour la fonction h.
6. Proposer un prototype pour une fonction en C qui calcule cette fonction de hachage (type de la valeur renvoyée, argument(s) et leur(s) type(s)).

❑ Exercice : *type B*

Pour un entier n quelconque, le but de l'exercice est de programmer en C une recherche en *backtracking* d'une solution au problème du placement des entiers $1, \dots, n^2$ entiers dans un carré de côté n afin de former un carré magique (c'est à dire un carré dont la somme des lignes, colonnes ou diagonales est la même)

1. Vérifier que dans le cas $n = 3$ le carré suivant est une solution :

2	7	6
9	5	1
4	3	8

2. Quelle serait la somme de chacune des lignes, colonnes ou diagonales dans le cas $n = 4$?
3. Donner l'expression de la somme de chacune des lignes, colonnes, ou diagonales pour un entier n quelconque.

   Afin d'implémenter la résolution en C, on propose de linéariser le carré en le représentant par un tableau à une seule dimension. Le carré ci-dessus est par exemple représenté par :
   `int carre[9] = {2, 7, 6, 9, 5, 1, 4, 3, 8}`
4. Donner l'expression de l'indice d'un élément situé en ligne i, colonne j dans le tableau linéarisé.
5. Inversement, donner la ligne et la colonne dans le carré initial d'un élément situé à l'indice i dans le tableau linéarisé.

   On représente par l'entier 0 une case non encore rempli du tableau, par exemple :
   `int carre[9] = {1, 8, 4, 0, 0, 0, 0, 0, 0}` représente un carré ayant simplement la première ligne complète
6. Ecrire une fonction `bool valide_ligne(int carre[], int lig, int size, int somme)` qui vérifie que la ligne `lig` d'un carré en cours de construction est encore valide, c'est le cas si cette ligne contient un zéro (car elle est alors incomplète) ou si elle ne contient aucun zéro et que sa somme est égale à la variable `somme` fournie en argument.

   On suppose écrites les fonctions correspondantes pour les colonnes et les lignes de même qu'une fonction : `bool valide_carre(int carre[], int size, int somme)` qui vérife qu'un carré en cours de construction est encore valide.
7. Ecrire la fonction permettant de rechercher par backtracking un carré magique solution du problème posé. On pourra passer en argument à cette fonction un tableau de booléens de taille n^2 dont l'élément d'indice i indique si l'entier i a déjà été placé ou non dans le carré.