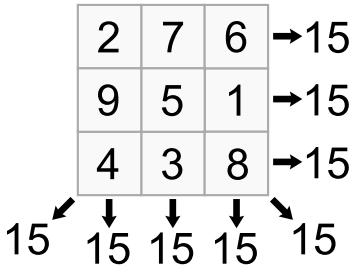
Fiche N°8 Cryptage

Dans ce TD vous devez décrire les fonctions et Main comme pour les algorithmes. Vous les implanterez en C# en TP

I. Carrés magiques

1. Écrire la fonction **estCarreMagique** » qui vérifie si un tableau NxN est un carré magique c'est-à-dire si les sommes sur chaque ligne, chaque colonne et sur chaque diagonale principale sont égales.
   Exemple :

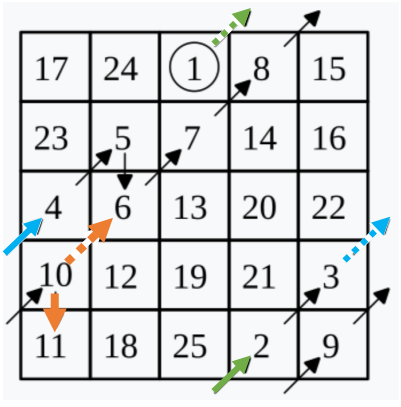


2. Il existe un algorithme simple pour construire un tel tableau quel que soit N impair.

Algorithme « Méthode siamoise » *(https://fr.wikipedia.org/wiki/Carré_magique_(mathématiques)*

- Placer le 1 au milieu de la première ligne
- Se décaler d'une case en diagonale à droite vers le haut.
  - Si ce déplacement amène à une case déjà remplie (flèche orange en pointillé), se déplacer d'une case vers le bas (flèche orange)
  - Si ce déplacement sort du carré en haut (flèche verte en pointillé), se placer sur la case la plus basse de la colonne de droite (flèche verte)
  - Si ce déplacement sort du carré à droite, (flèche bleue en pointillé) se placer sur la case la plus à gauche de la ligne du dessus (flèche bleue)
- Remplir la case avec le nombre suivant
- Recommencer à l'étape 2

Exemple :



3. Quel est le problème lorsque l'on se trouve dans la case en haut à droite (15).
4. On peut montrer que la case en bas à gauche sera toujours remplie (11) avant celle en haut à droite (15). En quoi ceci règle-t-il le problème ?

5. Écrire la fonction **creerCarreMagique** qui prend en paramètre un entier N et retourne un carré magique.

II. Cryptage amélioré

Pour compliquer le cryptage, on va utiliser une grille de codage/décodage. Le principe est simple. La GRILLE est donnée sous forme d'un tableau 27x27 de chaines de 3 caractères où les lignes et les colonnes sont indexées par l'alphabet de 'a' à 'z' plus le caractère ' '

	'a'	'b'	'c'	'd'	…	'w'	'x'	'y'	'z'	' '
'a'	"022"	"063"	"212"	"084"		"099"	…	"001"	…	"537"
'b'	"367"	…	…	…	…	…	…	…	…	"033"
'c'	…	…	…	…		…	…	…	…	…
…	…							…		
'x'	…	"058"	"413"	…	…	…	…	"322"	…	…
'y'	"057"	"412"	…	…		…	"541"	…	…	…
'z	…	…	…	…	…	…	…	…	…	"002"
' '	…	…	…	…		…	…	…	"036"	"698"

Chaque case représente un code pour un 2-gramme (un « mot » formé de deux caractères).
Exemple : "*aa*" est codé par "022", "*ab*" par "063", "*ba*" par "367", "*xy*" par "322"…

Le cryptage est alors très simple. Il suffit de prendre les lettres du message deux par deux et de cherche le code correspondant au 2-gramme directement dans le tableau.

Exemple : Codons «b a ba» *// S'il y a un nombre impair, on ajoute un espace à la fin*

- On prend les lettres deux à deux
  - On code "b " → GRILLE[1,26] → "033"
  - On code "a " → GRILLE[0,26] → "537"
  - On code "ba"→ GRILLE[0,1] → "367"
- Le mot codé : «033537367»

1. Quelle est la longueur LC du code d'un texte de longueur L
2. Sachant que le code ASCII de 'a' est 97, comment trouve-t-on sa ligne (et sa colonne) dans GRILLE ?
3. Écrire une fonction **donnerIndice** qui pour n'importe quel caractère C de l'alphabet donne son indice dans la grille
4. Soit un 2-gramme GG représenté par un tableau de 2 caractères. Quelle est la case GRILLE[i,j] correspondant à son codage ?
5. Écrire une fonction **code2gramme** qui retourne le codage d'un texte de longueur L donné en argument
6. Écrire une fonction **décode2gramme** qui retourne le décodage d'un texte de longueur LC donné en argument
7. Écrire un algorithme principal Main qui demande à l'utilisateur un texte à coder. Le texte codé est alors affiché. Il décodera ce texte et l'affichera (pour vérifier que votre décodage fonctionne !) *// On suppose que l'utilisateur donne bien un texte respectant les préconditions des différentes fonctions*

III. Cryptage à clé construite

Le problème du cryptage précédent est que la grille doit être connue du rédacteur et du lecteur. Il faut donc la transférer du rédacteur vers le lecteur avec risque d'interception. De plus, il est donc difficile de la changer en cas de nécessité. Pour remédier à cela on va utiliser un carré magique C de 27x27 où le 1 initial sera placer n'importe où dans la grille initiale. De fait, sa place $[i_0, j_0]$ sera la clé du (dé)codage. Le rédacteur n'aura qu'à transmettre cette clé au lecteur.
De plus, si on décide d'ajouter des lettre reconnues (par exemple les chiffres), il suffira de d'augmenter la taille du carré magique.
Dans la suite, on supposera que le nombre *Alphabet* de lettres reconnues est toujours impair.

1. Écrire une fonction **créerGrille** qui prend en paramètre la taille *Alphabet* de la grille et la position initiale du 1 et renvoie un tableau Alphabet X Alphabet correspondant à la grille de (dé)codage. .

   // Petite aide : Pour convertir un entier compris entre 0 et 999, on pourra s'inspirer du code C# suivant (fichier intToTexte.cs sur Moodle) :

   ```
   int number = 42;
   int taille = 3;
   string intString3 = number.ToString($"D{taille}");
   Console.WriteLine("La chaîne de caractères convertie est : " + intString3);
   ```

Le problème est que si on augmente le nombre de lettres reconnues, il peut y avoir un « trou ». En effet, par exemple, les chiffres '0', '1' … '9' ont un code ACSII compris entre 48 et 59 alors que 'A' a le code ACSII 65. Il faudra donc toujours ne prendre que des caractères qui se suivent dans la table ACSII. Dans la suite on prendra les lettres entre de '0' à 'Z' plus les lettres ' ' et '.'

2. Modifier la fonction **donnerIndice** en conséquence.
3. Modifier si nécessaire la fonction **Code2gramme** qui retourne le codage un texte de longueur L donné en argument
4. Modifier si nécessaire la fonction **Décode2gramme** qui retourne le décodage un texte de longueur LC donné en argument
5. Écrire un algorithme principal Main qui demande à l'utilisateur la clé de codage puis un texte à coder. Le texte codé est alors affiché. Il décodera ce texte et l'affichera (pour vérifier que votre décodage fonctionne !) *// On suppose que l'utilisateur donne une clé et un texte respectant les préconditions des différentes fonctions*

IV. Cryptage à deux phases

Pour encore renforcer le cryptage on va utiliser deux codages en cascade. Un texte sera d'abord codé par un codage Jules César puis seulement par un codage à clé construite $[i_0, j_0]$. On prend comme décalage le modulo *Alphabet* de $i_0 + j_0$

1. Écrire un algorithme principal Main qui demande à l'utilisateur la clé de codage puis un texte à coder. Le texte codé est alors affiché. Il décodera ce texte et l'affichera (pour vérifier que votre décodage fonctionne !) *// On suppose que l'utilisateur donne une clé et un texte respectant les préconditions des différentes fonctions*