Correction DS 5 - Math/Info

Exercice 1. Montrer que la matrice $P = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$ est inversible et calculer son inverse.

Correction 1. On applique la formule du pivot de Gauss, on obtient

$$\frac{1}{2}\begin{pmatrix} 1 & -1 & 1 \\ 1 & 1 & -1 \\ -1 & 1 & 1 \end{pmatrix}$$

Exercice 2. Soient $A = \begin{pmatrix} -1 & 5 & -3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$ et $P = \begin{pmatrix} 9 & 1 & -1 \\ -3 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$

1. Résoudre, pour tout $\lambda \in \mathbb{R}$, le système : $(A - \lambda I_3)X = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$ avec $X = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$.
2. Soit $Q = \begin{pmatrix} 1 & -2 & 1 \\ 3 & 10 & 3 \\ -4 & -8 & 12 \end{pmatrix}$. Calculer PQ.
3. En déduire que P est inversible et donner P^{-1} en fonction de Q.
4. Calculer $P^{-1}AP$.
5. En déduire que $A^n = PT^nP^{-1}$ pour tout $n \in \mathbb{N}$ où T est une matrice triangulaire à déterminer.
6. Montrer que pour tout $n \in \mathbb{N}$,
$$T^n = \begin{pmatrix} (-3)^n & 0 & 0 \\ 0 & 1 & -n \\ 0 & 0 & 1 \end{pmatrix}$$
7. On considère une suite $(u_n)_{n\in\mathbb{N}}$ définies par : $\begin{cases} u_0 = 0,\ u_1 = 0,\ u_2 = 1 \\ \forall n \in \mathbb{N},\ u_{n+3} = -u_{n+2} + 5u_{n+1} - 3u_n. \end{cases}$
   (a) On pose : $X_n = \begin{pmatrix} u_{n+2} \\ u_{n+1} \\ u_n \end{pmatrix}$. Donner la relation qui lie X_{n+1}, X_n et A pour tout $n \in \mathbb{N}$.
   (b) En déduire que pour tout $n \in \mathbb{N}$: $X_n = A^nX_0$.
   (c) En déduire l'expression explicite de la suite $(u_n)_{n\in\mathbb{N}}$.

Correction 2.

1. **Résoudre, pour tout $\lambda \in \mathbb{R}$, le système : $(\mathbf{A} - \lambda\mathbf{I_3})\mathbf{X} = \mathbf{0_{31}}$ avec $\mathbf{X} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$:**

   En utilisant la méthode du pivot de Gauss, on obtient que le système est équivalent au système échelonné suivant
$$\begin{cases} x - \lambda y = 0 \\ y - \lambda z = 0 \\ (\lambda-1)^2(\lambda+3)z = 0 \end{cases}$$
   On va donc devoir diviser par $(\lambda - 1)^2(\lambda + 3)$ et il faut donc étudier des cas selon la valeur de λ.

- Cas 1 : si $\lambda \notin \{-3, 1\}$:
  Le système est alors un système de rang 3, c'est donc un système de Cramer et il admet une unique solution qui est $\boxed{\mathcal{S}_{\lambda\notin\{-3,1\}} = \left\{\begin{pmatrix}0\\0\\0\end{pmatrix}\right\}.}$
- Cas 2 : si $\lambda = -3$:
  Le système est alors équivalent au système échelonné suivant : $\begin{cases} x + 3y = 0 \\ y + 3z = 0 \end{cases}$
  C'est un système de rang 2 dont les deux inconnues principales sont x et y et l'inconnue secondaire est z. On obtient que $\boxed{\mathcal{S}_{\lambda=-3} = \{(9z, -3z, z),\ z \in \mathbb{R}\}.}$
- Cas 3 : si $\lambda = 1$:
  Le système est alors équivalent au système échelonné suivant : $\begin{cases} x - y = 0 \\ y - z = 0 \end{cases}$
  C'est un système de rang 2 dont les deux inconnues principales sont x et y et l'inconnue secondaire est z. On obtient que $\boxed{\mathcal{S}_{\lambda=1} = \{(z, z, z),\ z \in \mathbb{R}\}.}$

2. Le calcul matriciel donne $PQ = 16I_3$.
3. Donc P est inversible, d'inverse
$$P^{-1} = \frac{1}{16}Q$$
4. Le calcul de deux produits matriciels donnent que : $\boxed{P^{-1}AP = \begin{pmatrix}-3 & 0 & 0\\ 0 & 1 & -1\\ 0 & 0 & 1\end{pmatrix}.}$ On note T cette matrice triangulaire supérieure.
5. Comme $T = P^{-1}AP$, on a : $\boxed{A = PTP^{-1}}$ car $T = P^{-1}AP \Leftrightarrow PT = AP \Leftrightarrow PTP^{-1} = A$ en utilisant le fait que $PP^{-1} = P^{-1}P = I_3$. Un raisonnement par récurrence permet de montrer que pour tout $n \in \mathbb{N}$: $\boxed{A^n = PT^nP^{-1}.}$
6. Pour calculer T^n comme c'est une matrice triangulaire supérieure on va pouvoir utiliser la formule du binôme de Newton.
   - ⋆ On a : $T = \begin{pmatrix}-3 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix} + \begin{pmatrix}0 & 0 & 0\\ 0 & 0 & -1\\ 0 & 0 & 0\end{pmatrix} = B + C$ avec B la matrice diagonale et C la matrice nilpotente.
   - ⋆ Pour tout $k \in \mathbb{N}$, on a : $B^k = \begin{pmatrix}(-3)^k & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}$.
     De plus on a : $C^2 = 0_3$ et ainsi pour tout $k \geq 2$: $C^k = 0_3$.
   - ⋆ Vérifions que les matrices B et C commutent bien entre elles. On a : $BC = \begin{pmatrix}0 & 0 & 0\\ 0 & 0 & -1\\ 0 & 0 & 0\end{pmatrix}$ et $CB = \begin{pmatrix}0 & 0 & 0\\ 0 & 0 & -1\\ 0 & 0 & 0\end{pmatrix}$. Donc $BC = CB$ et les matrices sont bien commutatives.

⋆ On peut donc appliquer la formule du binôme de Newton et on obtient que pour tout $n \in \mathbb{N}$:

$$\begin{aligned}
T^n &= (B+C)^n = \sum_{k=0}^{n} \binom{n}{k} C^k B^{n-k} \\
&= \sum_{k=0}^{1} \binom{n}{k} C^k B^{n-k} + \sum_{k=2}^{n} \binom{n}{k} C^k B^{n-k} \\
&= \binom{n}{0} C^0 B^n + \binom{n}{1} C^1 B^{n-1} \qquad \text{car } C \text{ est nilpotente} \\
&= \begin{pmatrix} (-3)^n & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -n \\ 0 & 0 & 0 \end{pmatrix} \\
&= \boxed{\begin{pmatrix} (-3)^n & 0 & 0 \\ 0 & 1 & -n \\ 0 & 0 & 1 \end{pmatrix}}.
\end{aligned}$$

7. **On considère une suite définie par $\mathbf{u_0 = 0},\ \mathbf{u_1 = 0},\ \mathbf{u_2 = 1}, \forall \mathbf{n} \in \mathbb{N},\ \mathbf{u_{n+3} = -u_{n+2} + 5u_{n+1} - 3u_n}$.**

   (a) **On pose pour tout $\mathbf{n} \in \mathbb{N}$: $\mathbf{X_n} = \begin{pmatrix} u_{n+2} \\ u_{n+1} \\ u_n \end{pmatrix}$. Donner la relation qui lie $\mathbf{X_{n+1}}$, $\mathbf{X_n}$ et $\mathbf{A}$ pour tout $\mathbf{n} \in \mathbb{N}$:**
   Le calcul donne que : $\boxed{X_{n+1} = AX_n}$ pour tout $n \in \mathbb{N}$ en utilisant la définition de la suite.

   (b) **En déduire que pour tout $\mathbf{n} \in \mathbb{N}$: $\mathbf{X_n = A^n X_0}$:**
   Il s'agit ici de faire une récurrence. Montrons par récurrence sur $n \in \mathbb{N}$ la propriété : $\mathcal{P}(n) :\ X_n = A^n X_0$.
   - Initialisation : pour $n = 0$:
     D'un côté, on a : X_0 et de l'autre côté, on a : $A^0 X_0 = I_3 X_0 = X_0$. Ainsi, $\mathcal{P}(0)$ est vraie.
   - Hérédité : soit $n \in \mathbb{N}$. On suppose la propriété vraie à l'ordre n, montrons qu'elle est vraie à l'ordre $n+1$. Par définition de la suite, on a : $X_{n+1} = AX_n$. Puis par hypothèse de récurrence, on sait que : $X_n = A^n X_0$. Ainsi, on obtient bien que : $X_{n+1} = A^{n+1} X_0$. Ainsi, $\mathcal{P}(n+1)$ est vérifiée.
   - Conclusion : il résulte du principe de récurrence que $\boxed{\forall n \in \mathbb{N}, \quad X_n = A^n X_0.}$

   (c) **En déduire l'expression explicite de la suite $(\mathbf{u_n})_{\mathbf{n} \in \mathbb{N}}$:**

   On peut calculer $A^n = PT^nP^{-1}$ pour tout $n \in \mathbb{N}$. En faisant deux produits matriciels, on obtient que pour tout $n \in \mathbb{N}$: $\boxed{A^n = \frac{1}{16} \begin{pmatrix} 4n+7+9(-3)^n & -18(-3)^n+8n+18 & 9(-3)^n-12n-9 \\ (-3)^{n+1}+4n+3 & -2(-3)^{n+1}+8n+10 & (-3)^{n+1}-12n+3 \\ (-3)^n+4n-1 & -2(-3)^n+8n+2 & (-3)^n-12n+15 \end{pmatrix}}$.

   En utilisant les conditions initiales, on sait que $X_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$. En calculant $A^n X_0$, on obtient que

$$\forall n \in \mathbb{N}, \quad \begin{aligned} u_{n+2} &= 9(-3)^n + 4n + 7 \\ u_{n+1} &= (-3)^{n+1} + 4n + 3 \\ u_n &= \boxed{(-3)^n + 4n - 1.} \end{aligned}$$

On remarque de plus que l'on retrouve bien u_{n+1} et u_{n+2} à partir de l'expression de u_n en remplaçant tous les n par des $n+1$ ou des $n+2$.

Exercice 3. On se place dans l'espace $\mathbb{R}^3$ et on considère le plan $\mathcal{P}$ d'équation cartisenne :

$$\mathcal{P} : x + 2y - z = 1$$

Pour tout ce problème, on fixe un point $A = (\alpha, \beta, \gamma)$ et on note $H(\lambda, \mu, \nu)$ son projeté orthogonal sur $\mathcal{P}$. Le but de ce problème est de déterminer (λ, μ, ν) en fonction de (α, β, γ)

1. Donner un vecteur normal à $\mathcal{P}$.
2. Déterminer une équation paramétrique de la droite $\mathcal{D}$ passant par A et orthogonale à $\mathcal{P}$.
3. Montrer que le plan $\mathcal{P}_2$ d'équation $\mathcal{P}_2 : -2x + y = \beta - 2\alpha$ contient la droite $\mathcal{D}$.
   On admet que l'on peut montrer de la même manière que le plan $\mathcal{P}_3$ d'équation $\mathcal{P}_3 : x + z = \alpha + \gamma$ contient la droite $\mathcal{D}$.
4. Justifier que $\mathcal{D} = \mathcal{P}_2 \cap \mathcal{P}_3$.
5. En déduire que les coordonnées de H vérifient un système linéaire qu'on peut écrire sous forme matricielle sous la forme :

   $$M_1 \begin{pmatrix} \lambda \\ \mu \\ \nu \end{pmatrix} = M_2 \quad \text{où } M_1 = \begin{pmatrix} 1 & 2 & -1 \\ -2 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \text{ et } M_2 \in M_{3,1}(\mathbb{R}) \text{ est une matrice colonne à déterminer.}$$

   M_2 s'exprimera en fonction de α, β, γ.
   On admet que M_1 est inversible et que

   $$M_1^{-1} = \frac{1}{6} \begin{pmatrix} 1 & -2 & 1 \\ 2 & 2 & 2 \\ -1 & 2 & 5 \end{pmatrix}$$

6. En déduire que

   $$\begin{pmatrix} \lambda \\ \mu \\ \nu \end{pmatrix} = P_1 \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} + P_2 \text{ où } P_1 = \frac{1}{6} \begin{pmatrix} 5 & -2 & 1 \\ -2 & 2 & 2 \\ 1 & 2 & 5 \end{pmatrix} \text{ et } P_2 = \frac{1}{6} \begin{pmatrix} 1 \\ 2 \\ -1 \end{pmatrix}$$

   On a ainsi répondu à la question principale. Deux questions supplémentaires vous sont proposées pour voir l'interaction entre la géométrie et les matrices.
7. Calculer P_1^2 et $P_1 P_2$
8. En déduire que

   $$P_1 \begin{pmatrix} \lambda \\ \mu \\ \nu \end{pmatrix} + P_2 = \begin{pmatrix} \lambda \\ \mu \\ \nu \end{pmatrix}$$

   et en expliquer la signification géométrique.

Correction 3.

1. Un vecteur normal à P est par exemple $(1, 2, -1)$
2. On obtient alors une équation de la drroite $\mathcal{D}$:

   $$\mathcal{D} : \begin{cases} x = & \alpha + t \\ y = & \beta + 2t \\ z = & \gamma - t \end{cases} \qquad t \in \mathbb{R}$$

3. Soit $(x, y, z) \in \mathcal{D}$. Il existe donc $t \in \mathbb{R}$ tel que

   $$(x, y, z) = (\alpha + t, \beta + 2t, \gamma - t).$$

   En remplacant dans l'équation de $\mathcal{P}_2$ on obtient

   $$-2(\alpha + t) + \beta + 2t = -2\alpha - 2t + \beta + 2T = -2\alpha + \beta$$

   Ainsi $(x, y, z) \in \mathcal{P}_2$

$$\boxed{\mathcal{D} \subset \mathcal{P}_2}$$

4. $\mathcal{P}_2$ et $\mathcal{P}_3$ sont deux plans non parallèles (coefficients non proportionnels) donc se coupent le long d'une droite. De plus, $\mathcal{D} \subset \mathcal{P}_2 \cap \mathcal{P}_3$ d'après la question précédente. Donc

$$\mathcal{D} = \mathcal{P}_2 \cap \mathcal{P}_3$$

5. Par définition $H = \mathcal{P} \cap D$. Donc d'après la question précédente

$$H = \mathcal{P} \cap \mathcal{P}_2 \cap \mathcal{P}_3$$

Donc les coordonnées de H vérifient les équations des trois plans :

$$\begin{cases} \lambda + 2\mu - \nu = 1 \\ -2\lambda + \mu = \beta - 2\alpha \\ \mu + \nu = \alpha + \gamma \end{cases}$$

Ce système se met sous forme matricielle selon

$$\begin{pmatrix} 1 & 2 & -1 \\ -2 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} \lambda \\ \mu \\ \nu \end{pmatrix} = \begin{pmatrix} 1 \\ \beta - 2\alpha \\ \alpha + \gamma \end{pmatrix}$$

On obtient donc $M_2 = \begin{pmatrix} 1 \\ \beta - 2\alpha \\ \alpha + \gamma \end{pmatrix}$

6. Comme M_1 est inversible on a

$$\begin{pmatrix} \lambda \\ \mu \\ \nu \end{pmatrix} = M_1^{-1} M_2$$

On fait le produit matriciel on obtient

$$M_1^{-1} M_2 = \frac{1}{6} \begin{pmatrix} 1 - 2\beta + 5\alpha + \gamma \\ 2 + 2\beta - 2\alpha + 2\gamma \\ -1 + 2\beta - \alpha + 5\gamma \end{pmatrix}$$

et on a

$$\frac{1}{6} \begin{pmatrix} 1 - 2\beta + 5\alpha + \gamma \\ 2 + 2\beta - 2\alpha + 2\gamma \\ -1 + 2\beta + \alpha + 5\gamma \end{pmatrix} = \frac{1}{6} \begin{pmatrix} -2\beta + 5\alpha + \gamma \\ 2\beta - 2\alpha + 2\gamma \\ 2\beta + \alpha + 5\gamma \end{pmatrix} + \frac{1}{6} \begin{pmatrix} 1 \\ 2 \\ -1 \end{pmatrix}$$

Enfin

$$\frac{1}{6} \begin{pmatrix} 5\alpha - 2\beta + \gamma \\ -2\alpha + 2\beta + 2\gamma \\ +\alpha + 2\beta + 5\gamma \end{pmatrix} = \frac{1}{6} \begin{pmatrix} 5 & -2 & 1 \\ -2 & 2 & 2 \\ 1 & 2 & 5 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}$$

7. Les calculs matriciels montrent que $P_1^2 = P_1$ et $P_1 P_2 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$

8. On a donc en utilisant Q6

$$P1 \begin{pmatrix} \lambda \\ \mu \\ \nu \end{pmatrix} = P_1^2 \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} + P_1 P_2$$

Donc

$$P1 \begin{pmatrix} \lambda \\ \mu \\ \nu \end{pmatrix} = P_1 \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}$$

De nouveau en remplacant dans l'égalité de la question 6 on obtient

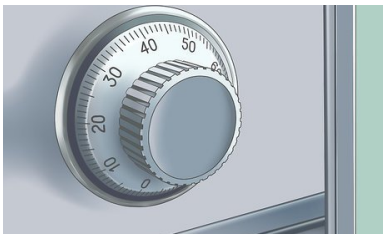
$$\begin{pmatrix}\lambda\\ \mu\\ \nu\end{pmatrix} = P1\begin{pmatrix}\lambda\\ \mu\\ \nu\end{pmatrix} + P_2$$

Ce qui donne l'égalité demandée.

Géométriquement, si on fait le projeté orthogonal d'un élément qu'on a déjà projeté, on ne fait rien.

Exercice 4.

Olivier voudrait voler le contenu du coffre fort de Yann, mais il ne connaît pas la combinaison qu'il a choisie. Il s'est cependant procuré le manuel du coffre, qui indique que la combinaison est nécessairement une suite de 3 à 8 entiers, tous compris entre 0 et 99, à rentrer dans l'ordre.



On note k la taille de la combinaison du coffre : on a donc $k \in [\![3, 8]\!]$.

Les questions sont indépendantes. À chaque fois, on demande de justifier (briévement) mais pas de faire les applications numériques, sauf mention explicite du contraire.

1. Combien de combinaisons sont possibles pour le coffre fort si on sait qu'elle est constituée de 4 nombres ?
2. Combien de combinaisons sont possibles en tout si on ne connaît pas k ?
3. Combien de combinaisons sont possibles si on sait que $k = 3$ et que les 3 nombres sont tous différents ?
4. Combien de combinaisons sont possibles si on sait que $k = 4$ et que la suite de nombres est strictement croissante ?
5. Olivier sait que Yann a choisi une combinaison à 5 nombres ($k = 5$), et il connaît aussi ses nombres préférés.
   (a) Combien de combinaisons possibles contiennent au moins une fois le nombre 77 ?
   (b) Combien de combinaisons contiennent exactement une fois chacun des nombres 1, 7, 17, 77 et 99 ? On donnera la valeur numérique.
6. **Python** On représente une combinaison en Python par une liste d'entiers.
   (a) Écrire une fonction `verifie(L)` qui prend en entrée une liste d'entiers `L`, et qui renvoie `True` si la liste correspond à une combinaison valide selon le manuel du coffre, et `False` sinon.
   (b) Écrire une fonction `strict_croissante(Comb)` qui prend en argument une liste `Comb` représentant une combinaison, et qui vérifie si elle est strictement croissante.

Correction 4.

1. C'est un choix avec ordre et répétition on a donc

$$100^4$$

possibilités. (Card($[\![0, 99]\!] = 100$)

2. Si on note $A_k = \{$choix possibles avec des codes à k chiffres $\}$ l'ensemble des codes possibles est alors $A = \cup_{k=3}^{8} A_k$. Comme l'union est disjointe on a

$$Card(A) = \sum_{k=3}^{8} 100^k$$

3. Si les nombres sont tous différents alors le choix se fait sans répétition. On a donc

$$\frac{100!}{(100-4)!} \text{ possibilités}$$

4. L'ensemble des suites de 4 nombres strictement croissant est en bijection avec l'ensemble des sous-ensemble à 4 éléments. On a donc
$$\binom{100}{4} \text{ possibilités}$$

5. (a) On passe au complémentaire. Les codes qui ne contiennent pas le nombre de 77 sont au nombre de 99^5 et donc ceux qui contiennent au moins une fois le nombre 77 sont au nombre de
$$100^5 - 99^5$$

   (b) Il suffit juste de trouver la place de ces 5 nombres, on en a donc
$$5! = 5 * 4 * 3 * 2 * 1 = 120$$

6. (a)
```
def verifie(L):
    if len(L) >8 or len(L)<3 :
        return False
    for i in L:
        if i>99 or i<0:
            return False
    return True
```

   (b)
```
def strict_croissante(L):
    for i in range(len(L)-1):
        if L[i+1]<L[i]:
            return False
    return True
```

Python

Exercice 5. On rappelle qu'une anagramme d'un mot est un mot obtenu en réarrangeant les lettres d'un autre mot. Par exemple :

`IMAGINER` et `MIGRAINE`

On ne supposera pas que les mots sont dans le dictionnaire français, ainsi

`IMAGINER` et `GRAINEMI`

seront aussi deux anagrammes valides dans ce problème.

Dans cette exercice, les chaînes de caractères ne contiendront que des lettres majuscules et pas de caractères spéciaux ou d'espaces.

Les fonctions `count` et `index` sont interdites. Les fonctions précédemment codées dans la copie, et même celles des questions admises, peuvent être utilisées dans les questions suivantes.

```
def anagramme_faux(s,t):
    if len(s)!=len(t):
        return(False)
    for lettre in s:
        if lettre not in t:
            return(False)
    return(True)
```

Trouver deux chaînes de caractères `ch1` et `ch2` qui ne sont pas anagrammes l'une de l'autre et qui pourtant vérifient `anagramme_faux(ch1,ch2)`—>`True` (je vous conseille de faire la question 2 avant de réfléchir à celle-ci pour comprendre le problème)

2. Détection si deux chaînes sont des anagrammes :
   (a) Écrire une fonction `NbAppar(ch,a)` qui prend en argument une chaine de caractères et un caractère et qui renvoie le nombre de fois que le caractère `a` apparait dans `ch`.

   Par exemple, `NbAppar('ANAGRAMME', 'A')` renvoie 3.

(b) Soient `ch1` et `ch2` deux chaînes de caractères de même longueur. Donner une condition nécessaire et suffisante portant sur le nombre d'apparitions de chaque lettre pour que ces deux chaînes de caractères soient des anagrammes.

(c) On considère la fonction :

```
def mystere(s):
    L=[]
    for l in s:
        if l not in L:
            L.append(l)
    return L
```

Que retourne `mystere('OLIVIER')` ?

(d) Écrire une fonction `VerfiAnag(ch1,ch2)` qui prend en argument deux chaînes de caractères et qui renvoie `True` si `ch1` et `ch2` sont des anagrammes et `False` sinon.

3. Dénombrement informatique des anagrammes d'un mot. Comme pour le dénombrement mathématiques, nous allons considérer *a priori* les lettres identiques comme différentes, puis nous allons enlever les doublons.

(a) Parmi les quatre fonctions suivantes, déterminer l'unique fonction qui prend en argument une chaîne de caractère `ch` et un caractère `a` et qui renvoie la liste chaînes de caractères obtenue en insérant à chaque position possible de `ch`.

Par exemple, pour `ch='BO'` et `a='A'`, on doit obtenir `['ABO', 'BAO', 'BOA']`. Pour `ch='BO'` et `a='B'`, on doit obtenir `['BBO', 'BBO', 'BOB']` (le premier et le deuxième 'BBO' correspondent respectivement à l'ajout du 'B' à la position 0 puis à la position 1)

Pour celles qui ne réalisent pas ce qui est demandé on donnera un contre exemple d'une chaîne `ch` et d'un caractère `a` et on donnera le résultat de la fonction.

```
def Inser1(ch, a) :
    L = []
    for k in range(len(ch)) :
        L.append(ch[:k]+a+ch[k:])
    return L
```

```
def Inser2(ch,a) :
    L = []
    for k in range(len(ch)+1) :
        L.append(ch[:k]+a+ch[k:])
    return L
```

```
def Inser3(ch,a) :
    L = []
    for k in range(len(ch)) :
        L.append(ch[:k]+a+ch[k+1:])
    return L
```

```
def Inser4(ch,a) :
    L = []
    for k in range(len(ch)+1) :
        L.append(ch[:k]+a+ch[k+1:])
    return L
```

(b) Utiliser la fonction d'insertion précédente pour écrire une fonction `InserListe(L,a)` qui prend en argument une liste de chaines de caractères et qui renvoie la liste des chaines de caractères obtenue en insérant à chaque position possible de chacune des chaines de caractères de `L`.

Par exemple, `InserListe(['OB', 'BO'], 'B')` renvoie `['BOB','OBB','OBB','BBO','BBO','BOB']`.

(c) Compléter la fonction `ListeAnag(ch)` qui prend en argument une chaîne de caractères et afin qu'elle renvoie la liste des anagrammes de la chaîne de caractères (éventuellement avec répétition).

Par exemple, `ListeAnag('BOB')` devra renvoyer `['BOB','OBB','OBB','BBO','BBO','BOB']`.

```
    def ListeAnag(ch):
        L=['']
        n=...
        for i in range(n):
            L=inserListe(... , ...)
        return
```

(d) Ecrire une fonction `SansRepet(L)` qui prend en argument une liste `L` pour et qui renvoie la même liste sans répétition.

Par exemple, `SansRepet(['BOB','OBB','OBB','BBO','BBO','BOB']` renvoie `['BOB','OBB','BBO']`.

(e) Utiliser les fonctions précédentes pour écrire une fonction `NbAnag(ch)` qui prend en argument une chaîne de caractères et qui renvoie le nombre d'anagrammes de ce mot.

(f) Quelle valeur va retourner `NbAnag('OLIVIER')` ? (à exprimer à l'aide de factorielle - ce n'est pas une question d'info)

Correction 5.

1. L'erreur se trouve dans le fait de ne pas compter le nombre de chaque lettre. Ainsi anagramme('OLIVIER','OLOV renvoie True alors que 'OLIVIER','OLOVIER' ne sont pas anagrammes l'un de l'autre.

2. (a)
```
def NbAppar(ch,a):
    c=0 #compteur
    for lettre in ch:
        if lettre==a:
            c=c+1
    return(c)
```

(b) Pour que deux chaines de même longueur soient anagrammes l'une de l'autre il faut et il suffit que chaque lettre de la première apparaisse exactement le meme nombre de fois dans les deux chaines.

(c)
```
def VerifAnag(ch1,ch2):
    if len(ch1)!=len(ch2):
        return False

    for lettre in ch1:
        if NbAppar(ch1,lettre)!=NbAppar(ch2,lettre):
            return False
    return True
```

(d) C'est la fonction 2. Voila les différents résultats pour chacune des fonctions, en prenant `ch='BOU'` et `a='A'`

```
inser1 ['ABOU', 'BAOU', 'BOAU']
inser2 ['ABOU', 'BAOU', 'BOAU', 'BOUA']
inser3 ['AOU', 'BAU', 'BOA']
inser4 ['AOU', 'BAU', 'BOA', 'BOUA']
```

(e)
```
def inserListe(L,a):
    K=[]
    for ch in L:
        K=K+inser(ch,a)
    return(K)
```

```
def listAnag(ch):
    K=['']
    for lettre in ch:
        K=inserListe(K,lettre)
    return(K)
```

(f)(g)
```
def SansRepet(L):
    L2=[]
    for l in L:
        if l not in L2:
            L2.append(l)
    return(L2)
```

(h)
```
def NbAnag(ch):
    L=listAnag(ch)
    L2=SansRepet(L)
    retur(len(L2))
```

(i) La fonction doit retourner $\frac{7!}{2!}$