Correction DS 5

Exercice 1. Soit f une fonction de $\mathbb{R}$ dans $\mathbb{R}$. On considère les trois propositions suivantes

$$P_1(f) : "\exists M \in \mathbb{R}, \forall x \in \mathbb{R}, f(x) < M"$$

$$P_2(f) : "\exists x \in \mathbb{R}, \exists y \in \mathbb{R}, f(x) < f(y)"$$

$$P_3(f) : "\forall x \in \mathbb{R}, \exists y \in \mathbb{R}^+, f(x) \geq f(y)"$$

1. Donner les négations de ces propositions
2. Dire si ces propositions sont vraies ou fausses pour les fonctions suivantes :

$$f \left| \begin{array}{ccc} \mathbb{R} & \to & \mathbb{R} \\ x & \mapsto & 1 \end{array} \right. , \quad g \left| \begin{array}{ccc} \mathbb{R} & \to & \mathbb{R} \\ x & \mapsto & \exp(x) \end{array} \right. , \quad h \left| \begin{array}{ccc} \mathbb{R} & \to & \mathbb{R} \\ x & \mapsto & \cos(x) \end{array} \right.$$

On justifiera, dans le cas où les propositions sont vraies, en donnant une valeur pour les variables quantifiées par le quantificateur $\exists$

Correction 1.

1.
$$NON(P_1(f)) : "\forall M \in \mathbb{R}, \exists x \in \mathbb{R}, f(x) \geq M"$$
$$NON(P_2(f)) : "\forall x \in \mathbb{R}, \forall y \in \mathbb{R}, f(x) \geq f(y)"$$
$$NON(P_3(f)) : "\exists x \in \mathbb{R}, \forall y \in \mathbb{R}^+, f(x) < f(y)"$$

2. Pour f
   - $P_1(f)$ est vrai, il suffit de prendre $M = 2$.
   - $P_2(f)$ est faux.
   - $P_3(f)$ est vrai, il suffit de prendre $y = 1$.

   Pour g
   - $P_1(g)$ est faux.
   - $P_2(g)$ est vrai, il suffit de prendre $x = 1$ et $y = 2$.
   - $P_3(g)$ est faux.

   Pour h
   - $P_1(h)$ est vrai il suffit de prendre $M = 2$.
   - $P_2(h)$ est vrai, il suffit de prendre $x = -\pi$ et $y = 0$.
   - $P_3(h)$ est vrai, il suffit de prendre $y = \pi$.

Exercice 2. Soit M la matrice :

$$M = \begin{pmatrix} -1 & -3 & 0 \\ 0 & 2 & 0 \\ 3 & 2 & 2 \end{pmatrix}$$

1. Résoudre le système $MX = \lambda X$ d'inconnue $X = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ où λ est un paramètre réel.
2. Calculer $(M - 2\,\mathrm{Id})^2$. Donner son rang.

3. Soit $e_1 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$, $e_2 = \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}$, et $e_3 = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}$. Exprimer Me_1, Me_3 en fonction de e_1, e_3.
4. Montrer qu'il existe $(\alpha, \beta) \in \mathbb{R}^2$ tel que $Me_2 = \alpha e_2 + \beta e_1$.
5. Soit $P = \begin{pmatrix} 0 & 1 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & -1 \end{pmatrix}$
   Montrer que P est inversible et calculer son inverse.
6. Soit $T = P^{-1}MP$. Calculer T.
7. En déduire l'expression de T^n en fonction de P, M et n. (La récurrence n'est pas exigée)
8. On pose $D = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & -1 \end{pmatrix}$ et $N = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$
   Vérifier que
   $$T = D + N \quad \text{et} \quad ND = DN$$
9. Calculer N^2
10. Montrer que $T^n = D^n + nND^{n-1}$.
11. Soit $(x_n)_{n\in\mathbb{N}}$, $(y_n)_{n\in\mathbb{N}}$ et $(z_n)_{n\in\mathbb{N}}$ trois suites vérifiant $x_0 = y_0 = z_0 = 1$ et pour tout $n \in \mathbb{N}$
    $$\begin{cases} x_{n+1} &= -x_n - 3y_n \\ y_{n+1} &= 2y_n \\ z_{n+1} &= 3x_n + 2y_n + 2z_n \end{cases}$$
    On considère $U_n = \begin{pmatrix} x_n \\ y_n \\ z_n \end{pmatrix}$
    (a) Déterminer une relation de récurrence entre U_{n+1}, U_n et M.
    (b) En déduire à l'aide d'une récurrence l'expression de U_n en fonction de M, n et U_0.
    (c) En déduire l'expression de x_n en fonction de n.

Correction 2.

1.
$$MX = \lambda X \iff \begin{pmatrix} -x - 3y \\ 2y \\ 3x + 2y + 2z \end{pmatrix} = \begin{pmatrix} \lambda x \\ \lambda y \\ \lambda z \end{pmatrix}$$
$$\begin{cases} -x & -3y & & = & \lambda x \\ & 2y & & = & \lambda y \\ 3x & +2y & +2z & = & \lambda z \end{cases} \iff \begin{cases} (-1-\lambda)x & -3y & & = & 0 \\ & (2-\lambda)y & & = & 0 \\ 3x & +2y & +(2-\lambda)z & = & 0 \end{cases}$$
En échangeant les lignes et les colonnes on peut voir que le système est déjà échelonné. $L_3 \leftarrow L_1, L_2 \leftarrow_3, L_1 \leftarrow L_2, C_3 \leftarrow C_1, C_2 \leftarrow C_3, C_1 \leftarrow C_2$
$$MX = \lambda X \iff \begin{cases} (2-\lambda)z & +3x & +2y & = & 0 \\ & (-1-\lambda)x & -3y & = & 0 \\ & & (2-\lambda)y & = & 0 \end{cases}$$
Si $\lambda \notin \{-1, 2\}$ alors le système est de rang 3, il est donc de Cramer et l'unique solution est

$$\boxed{\mathcal{S} = \left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \right\}}$$

Si $\lambda = -1$, le système est équivalent à

$$\left\{ \begin{array}{rrrcl} 3z & +3x & +2y = & 0 \\ & -3y & = & 0 \\ & 3y & = & 0 \end{array} \right. \iff \left\{ \begin{array}{rl} z & = -x \\ y & = 0 \end{array} \right.$$

Le système est de rang 2. L'ensemble des solutions est

$$\boxed{\mathcal{S} = \left\{ \begin{pmatrix} x \\ 0 \\ -x \end{pmatrix}, x \in \mathbb{R} \right\}}$$

Si $\lambda = 2$, le système est équivalent à

$$\left\{ \begin{array}{rrcl} 3x & +2y & = & 0 \\ -3x & -3y & = & 0 \\ & 0 & = & 0 \end{array} \right. \iff \left\{ \begin{array}{rrl} x & & = 0 \\ & y & = 0 \end{array} \right.$$

Le système est de rang 2. L'ensemble des solutions est

$$\boxed{\mathcal{S} = \left\{ \begin{pmatrix} 0 \\ 0 \\ z \end{pmatrix}, z \in \mathbb{R} \right\}}$$

2. $M - \mathrm{Id} = \begin{pmatrix} -3 & -3 & 0 \\ 0 & 0 & 0 \\ 3 & 2 & 0 \end{pmatrix}$

   Donc

$$\boxed{(M - \mathrm{Id})^2 = \begin{pmatrix} 9 & 9 & 0 \\ 0 & 0 & 0 \\ -9 & -9 & 0 \end{pmatrix}}$$

   Le système associé est

   $\left\{ \begin{array}{rrl} 9x & +9y & = 0 \\ & & 0 = 0 \\ -9x & -9y & = 0 \end{array} \right. \iff \left\{ \begin{array}{rl} x + y & = 0 \end{array} \right.$ Il est de rang 1. Donc

$$\boxed{(M - \mathrm{Id})^2 \text{ est de rang } 1}$$

3. Le calcul montre que $Me_1 = 2e_1$ et $Me_3 = -e_3$

4. Le calcul montre que $Me_2 = \begin{pmatrix} 2 \\ -2 \\ 1 \end{pmatrix} = 2 \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ \\ 1 \end{pmatrix} 2e_2 + e_1$

   Ainsi on peut prendre

$$\boxed{\alpha = 2 \text{ et } \beta = -1}$$

5. On considère la matrice augmentée : $\left(\begin{array}{ccc|ccc} 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 & 0 & 1 \end{array}\right)$

$L_3 \leftrightarrow L_1$ donnent

$$\left(\begin{array}{ccc|ccc} 1 & 0 & -1 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \end{array}\right)$$

$L_3 \leftarrow L_3 + L_2$ donne

$$\left(\begin{array}{ccc|ccc} 1 & 0 & -1 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right)$$

$L_2 \leftarrow -L_2$ donne

$$\left(\begin{array}{ccc|ccc} 1 & 0 & -1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right)$$

Enfin $L_1 \leftrightarrow L_1 + L_3$ donne

$$\left(\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right)$$

$$\boxed{P \text{ est inversible d'inverse } \left(\begin{array}{ccc} 1 & 1 & 1 \\ 0 & -1 & 0 \\ 1 & 1 & 0 \end{array}\right)}$$

6. Le calcul donne

$$\boxed{T = \left(\begin{array}{ccc} 2 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & -1 \end{array}\right)}$$

(sur une copie, le produit intermédiaire MP serait apprécié)

7. On pose $P(n)$: "$T^n = P^{-1}M^nP$"

Initialisation $T^1 = T$ et $P^{-1}M^1P = P^{-1}MP = T$ d'après la définition de T. Donc $P(1)$ est vrai.

Hérédité On suppose qu'il existe $n \in \mathbb{N}$ tel que $P(n)$ soit vraie. On a alors

$$(T)^{n+1} = T^nT$$

et donc par Hypothése de récurrence :

$$\begin{aligned} T^{n+1} &= (P^{-1}M^nP)(P^{-1}MP) \\ &= (P^{-1}M^nPP^{-1}MP) \\ &= (P^{-1}M^n \operatorname{Id} MP) \\ &= (P^{-1}M^nMP) \\ &= (P^{-1}M^{n+1}P) \end{aligned}$$

Conclusion $P(n)$ est vraie pour tout n.

8. On a $T = \left(\begin{array}{ccc} 2 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & -1 \end{array}\right) = \left(\begin{array}{ccc} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & -1 \end{array}\right) + \left(\begin{array}{ccc} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{array}\right)$

et

$$DN = \left(\begin{array}{ccc} 0 & 2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{array}\right) = ND$$

9. $N^2 = 0_3$
10. Solution 1 : On peut appliquer le binome de Newton à $T = D + N$ car D et N commutent. On a alors

$$T^n = \sum_{k=0}^{n} \binom{n}{k} N^k D^{n-k}$$

Comme pour tout $k \geq 2$, $N^2 = 0$ il reste dans cette somme seulement les termes $k = 0$ et $k = 1$. On obtient donc

$$\begin{aligned} T^n &= \binom{n}{0} N^0 D^{n-0} + \binom{n}{1} N^1 D^{n-1} \\ &= D^n + nND^{n-1} \end{aligned}$$

Solution 2 :
On pose $P(n)$: ´ $T^n = D^n + nD^{n-1}N$ ˇ

— Initialisation $T^1 = T$ et $D^1 + 1D^0N = D^1 + \mathrm{Id}\, N = D + N = T$ d'après la définition de D, N. Donc $P(1)$ est vrai.

— Hérédité On suppose qu'il existe $n \in \mathbb{N}$ tel que $P(n)$ soit vraie. On a alors

$$(T)^{n+1} = T^n T$$

et donc par Hypothése de récurrence :

$$\begin{aligned} T^{n+1} &= (D^n + nD^{n-1}N)(D + N) \\ &= D^n D + nD^{n-1}ND + D^n N + nD^{n-1}N^2 \end{aligned}$$

Comme $ND = DN$ on a $D^{n-1}ND = D^{n-1}DN = D^n N$. on a par ailleurs $N^2 = 0$ donc

$$\begin{aligned} T^{n+1} &= D^{n+1} + D^n N + nD^n N \\ &= D^{n+1} + (n+1)D^{(n+1)-1}N \end{aligned}$$

Ainsi la propriété est héréditaire.

— Conclusion $P(n)$ est vraie pour tout n.

11. Le système donne la relation suivante
    (a) (attention au sens entre M et U_n)

$$\boxed{U_{n+1} = MU_n}$$

    (b) Faire la récurrence pour montrer

$$\boxed{U_n = M^n U_0}$$

    (c) On a donc $U_n = \begin{pmatrix} 2^n & 2^n - 1 & 0 \\ 0 & 1 & 0 \\ -2^n + 1 & -2^n + 1 + n & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ On obtient

$$\boxed{x_n = 2^{n+1} - 1}$$

Exercice 3. Pour Noël, la 1BIO A a décidé d'organiser un échange de cadeaux au hasard au cours duquel chaque personne reçoit un cadeau d'une autre personne. Pour ce faire, tous les noms (étudiants et professeurs) sont placés dans une trousse et chaque personne tire au hasard le nom de la personne à qui elle offrira son cadeau. Si une personne tire son propre nom, elle en tire un autre puis remet son nom dans le chapeau. On propose d'étudier le nombre de façons possibles d'attribuer tous les cadeaux.

On désigne par $n \geqslant 2$ le nombre total de personnes et par a_n le nombre de façons possibles d'attribuer les n cadeaux sans que quelqu'un reçoive son propre cadeau. Le but de cet exercice est de déterminer une relation de récurrence vérifiée par a_n.

1. Déterminer a_2 et a_3.
2. On fixe $n \geqslant 4$. Soit x votre professeur de mathématiques. On désigne par y la personne qui reçoit le cadeau de x et par z la personne qui reçoit le cadeau de y. On exprimera les réponses aux trois questions suivantes en fonction de n, a_{n-1} et a_{n-2}.
   (a) Combien y a-t-il de choix possibles pour y ?
   (b) Si $z = x$, combien y a-t-il de façons possibles d'attribuer les n cadeaux ?
   (c) Si $z \neq x$, combien y a-t-il de façons possibles d'attribuer les n cadeaux ? (Indication : on pourra étudier le cas où y se désiste et x donne directement son cadeau à z.)
3. En déduire que $a_n = (n-1)\left(a_{n-1} + a_{n-2}\right)$ pour tout $n \geqslant 4$.
4. (a) Montrer que $\forall n \geq 2$

$$(n+1)\left(n!\sum_{k=0}^{n}\frac{(-1)^k}{k!} + (n+1)!\sum_{k=0}^{n+1}\frac{(-1)^k}{k!}\right) = (n+2)!\left(\sum_{k=0}^{n}\frac{(-1)^k}{k!}\right) + (n+1)(-1)^{n+1}$$

   (b) Montrer que $\forall n \geq 2$

$$\sum_{k=0}^{n}\frac{(-1)^k}{k!} = \left(\sum_{k=0}^{n+2}\frac{(-1)^k}{k!}\right) - (-1)^{n+1}\frac{n+1}{(n+2)!}$$

   (c) En déduire à l'aide d'une récurrence double que

$$a_n = n!\sum_{k=0}^{n}\frac{(-1)^k}{k!}$$

   pour tout $n \geqslant 2$. (on rappelle que par convention $0! = 1$)

Correction 3.

1. $a_2 = 1$ la seule façon de procéder est que la personne P_1 donne à P_2 et réciproquement. $a_3 = 2$: P_1 a deux choix puis : P_2 et P_3, puis les autres n'ont plus qu'un seul choix.
2. (a) Il y a $n-1$ choix pour y, tout le monde sauf x.
   (b) Si $z = x$, ensuite il faut choisir la distribution des cadeaux entre les $n-2$ autres personnes. Il y en a donc a_{n-2}.
   (c) Si $z \neq x$, on considère comme le suggère l'énoncé le cas où y se désiste. Il faut donc réaliser un secret santa, entre les $n-1$ personnes restantes, soit a_{n-1} possibilités.
3. D'après la question précédente, il y a $n-1$ façons de choisir y puis $a_n + an - 1$ façons d'attribuer les cadeaux une fois y choisi. Ainsi

$$\boxed{a_n = (n-1)(a_{n-1} + a_{n-2})}$$

4. Soit $P(n)$ la propriété " $a_n = n! \sum_{k=0}^{n} \frac{(-1)^k}{k!}$ "

   Initialistion : $P(2)$ et $P(3)$ sont vraies en effet :

$$\begin{aligned} 2! \sum_{k=0}^{2} \frac{(-1)^k}{k!} &= 2 * (\frac{1}{0!} + \frac{-1}{1!} + \frac{1}{2!}) \\ &= 2(1 - 1 + \frac{1}{2}) \\ &= 1 \\ &= a_2 \end{aligned}$$

   et

$$\begin{aligned} 3! \sum_{k=0}^{3} \frac{(-1)^k}{k!} &= 6 * (\frac{1}{0!} + \frac{-1}{1!} + \frac{1}{2!} + \frac{-1}{3!}) \\ &= 6(1 - 1 + \frac{1}{2} - \frac{1}{6}) \\ &= 6 * \frac{2}{6} \\ &= 2 \qquad\qquad = a_3 \end{aligned}$$

   Hérédité : On suppose qu'il existe $n \in [\![2, +\infty]\!]$ tel que $P(n)$ et $P(n+1)$ sont vraies et on va montrer $P(n+2)$. On a d'après la question 3

$$a_{n+2} = (n+1)(a_n + a_{n+1})$$

   et donc d'après l'hypothèse de récurrence :

$$a_{n+2} = (n+1) \left(n! \sum_{k=0}^{n} \frac{(-1)^k}{k!} + (n+1)! \sum_{k=0}^{n+1} \frac{(-1)^k}{k!} \right)$$

   Ainsi

$$\begin{aligned} a_{n+2} &= (n+1) \left(n! \sum_{k=0}^{n} \frac{(-1)^k}{k!} + (n+1)n! \sum_{k=0}^{n} \frac{(-1)^k}{k!} + (n+1)! \frac{(-1)^{n+1}}{(n+1)!} \right) \\ &= (n+1) \left((n!(1 + (n+1)) \sum_{k=0}^{n} \frac{(-1)^k}{k!} + (-1)^{n+1} \right) \\ &= (n+1)(n+2)n! \sum_{k=0}^{n} \frac{(-1)^k}{k!} + (n+1)(-1)^{n+1} \\ &= (n+2)! \sum_{k=0}^{n} \frac{(-1)^k}{k!} + (n+1)(-1)^{n+1} \end{aligned}$$

Or

$$\begin{aligned}\sum_{k=0}^{n}\frac{(-1)^k}{k!} &= \sum_{k=0}^{n+2}\frac{(-1)^k}{k!} - \left(\frac{(-1)^{n+1}}{(n+1)!}+\frac{(-1)^{n+2}}{(n+2)!}\right)\\ &= \sum_{k=0}^{n+2}\frac{(-1)^k}{k!} - (-1)^{n+1}\frac{n+2-1}{(n+2)!}\\ &= \sum_{k=0}^{n+2}\frac{(-1)^k}{k!} - (-1)^{n+1}\frac{n+1}{(n+2)!}\end{aligned}$$

Donc

$$\begin{aligned}a_{n+2} &= (n+2)!\left(\sum_{k=0}^{n+2}\frac{(-1)^k}{k!} - (-1)^{n+1}\frac{n+1}{(n+2)!}\right) + (n+1)(-1)^{n+1}\\ &= (n+2)!\sum_{k=0}^{n+2}\frac{(-1)^k}{k!} - (n+2)!(-1)^{n+1}\frac{n+1}{(n+2)!} + (n+1)(-1)^{n+1}\\ &= (n+2)!\sum_{k=0}^{n+2}\frac{(-1)^k}{k!}\end{aligned}$$

INFORMATIQUE

Exercice 4 (Anagrammes)**.**

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`

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

1. Dans le DM des vacances de Noël j'avais proposé une fonction qui permettait de vérifier si deux mots étaient des anagrammes l'un de l'autre. Il y avait malheureusement une erreur dans cette fonction. M. Lemanissier vous propose un exercice qui permet de combler cette erreur.

   Voici la fonction du DM :

```
def anagramme(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(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) 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)

```
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) ?

Correction 4.

1. L'erreur se trouve dans le fait de ne pas compter le nombre de chaque lettre. Ainsi anagramme('OLIVIER','OL 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)
```
def SansRepet(L):
    L2=[]
    for l in L:
        if l not in L2:
            L2.append(l)
    return(L2)
```

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

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