Correction du partiel du 12/11/2024

Remarques générales :

1. Soigner les preuves de terminaison, attention aux suites qui sont décroissantes ou strictement décroissantes, ou strictement décroissantes jusqu'au moment où elles stationnent etc.
2. Preuves de correction : attention aux récurrences, en particulier à l'hypothèse de récurrence : concerne-t-elle un objet fixé à l'avance ? Est-elle vraie pour tout objet d'un certain type ? Attention à bien rédiger une récurrence forte. Ne pas chercher à transformer les récurrences fortes en récurrences simples en mettant le $\forall$ dans l'assertion, rédiger simplement par récurrence forte.
3. Attention aux DL, limites, grands et petits o etc, il y en aura aux prochaines interros et à l'examen.
4. Attention à la complexité, et même si c'est bien compris, attention à bien justifier. Deux boucles peuvent être imbriquées ou pas, ça change la complexité, ne pas juste écrire « il y a deux boucles » : dans ce partiel il avait des cas avec deux boucles à la suite et d'autres cas avec deux boucles imbriquées. Pour simplifier une rédaction, vous pouvez majorer un peu brutalement, si vous voulez uniquement un grand O.
5. Suggestion de site web pour s'entraîner à écrire des algorithmes : `https://projecteuler.net/`. Conseil : recoder tous les algorithmes vus en cours en version récursive, pour s'entraîner. Ne pas hésiter à googler pour chercher des exercices d'algorithmique, on trouve beaucoup de choses par exemple `https://www.geeksforgeeks.org/recursion-practice-problems-solutions/`.

Correction de l'exercice 1

Comme $\frac{1}{n} + \frac{1}{n^2}$ tend vers zéro en $+\infty$, on peut utiliser le développement limité de $\sqrt{1+u}$ lorsque $u \to 0$, ce qui donne :

$$\begin{aligned}\sqrt{1+\frac{1}{n}+\frac{1}{n^2}} &= 1 + \frac{1}{2}\left(\frac{1}{n}+\frac{1}{n^2}\right) - \frac{1}{8}\left(\frac{1}{n}+\frac{1}{n^2}\right)^2 + o\left(\frac{1}{n^2}\right) \\ &= 1 + \frac{1}{2n} + \left(\frac{1}{2} - \frac{1}{8}\right)\frac{1}{n^2} + o\left(\frac{1}{n^2}\right)\end{aligned}$$

Donc $\sqrt{n^2+n+1} = n + \frac{1}{2} + \frac{3}{8n} + o\left(\frac{1}{n^2}\right)$.

De même, $\sqrt{n^2+n-1} = n + \frac{1}{2} - \frac{5}{8n} + o\left(\frac{1}{n^2}\right)$.

On en déduit que $u_n \sim \frac{1}{n}$.

Remarque : entraînez-vous de manière constante sur les DL, vous en aurez aux prochaines évaluations. Vous devez connaître vos DL classiques, sachant qu'on verra bientôt un chapitre sur l'approximation des valeurs des fonctions où il faudra connaître ses développements en série entière classiques.

Correction de l'exercice 2

Dans l'ensemble bien fait mais souvent mal justifié.

1. Somme des chiffres en base 10 :

```python
def somme_chiffres(n: int) -> int:
  if n < 10:
    return n
  return n % 10 + somme_chiffres(n//10)
```

— Terminaison : les appels se font sur des entiers ayant à chaque fois un chiffre de moins en base 10, jusqu'à arriver à un entier à un seul chiffre, moment où la récursivité s'arrête. Autre rédaction : les appels récursifs se font sur des entiers qui diminuent strictement donc passent en-dessous de 10 après un nombre fini d'appels.

— Correction : montrons la correction par récurrence sur le nombre de chiffres de l'entier. Si $k \in \mathbb{N}^*$, notons $\mathcal{P}(k)$ l'assertion « pour tout nombre n ayant k chiffres en base 10, `somme_chiffres(n)` est la somme des chiffres de n. »
Initialisation : $\mathcal{P}(1)$ est vraie : si l'entier n'a qu'un chiffre, on a $n = n\%10$ et $n//10 = 0$, donc la fonction renvoie n ce qui est bien le résultat voulu.
Hérédité : soit $k \in \mathbb{N}$. Supposons $\mathcal{P}(k)$. Prouvons $\mathcal{P}(k+1)$. Soit n un entier ayant $k+1$ chiffres en base 10. Écrivons la division euclidienne de n par 10 :

$$n = 10 \times (n//10) + (n\%10).$$

L'entier $n//10$ a k chiffres en base 10, ce sont les k premiers chiffres de n, et $n\%10$ est son dernier chiffre. D'après l'hypothèse de récurrence, `somme_chiffres(n//10)` est la somme des k premiers chiffres de n, et donc `somme_chiffres(n)` est la somme des chiffres de n. Ceci termine l'hérédité.

Remarque : on peut aussi rédiger par récurrence forte sur n, au lieu de faire une récurrence sur le nombre k de chiffres. Attention à ne pas faire de récurrence simple sur n.

2. Somme des valeurs paires d'une liste :

```
def nb_termes_pairs(L: list) -> int:
  if L == []:
    return 0
  if L[0] % 2 == 0:
    return 1 + somme_pairs(L[1:])
  else:
    return somme_pairs(L[1:])
```

— Terminaison : les appels se font sur des listes ayant une longueur diminuant de un à chaque fois, et la récursivité s'arrête pour une liste vide.

— Correction : pour $n \in \mathbb{N}$, notons $\mathcal{P}(n)$ l'assertion « pour tout liste L de longueur n, `somme_pairs(L)` est égal au nombre de termes pairs dans la liste. ». Montrons $\forall n \in \mathbb{N}, \mathcal{P}(n)$ par récurrence sur n. L'assertion $\mathcal{P}(0)$ est vraie. Soit $n \in \mathbb{N}$, supposons $\mathcal{P}(n)$, et montrons $\mathcal{P}(n+1)$. Soit L une liste de longueur $n+1$. Alors la liste $L[1:]$ est de longueur n et donc par hypothèse de récurrence appliquée à la liste $L[1:]$, le nombre `somme_pairs(L[:1])` est égal au nombre de termes pairs dans $L[:1]$, c'est-à-dire dans les n derniers termes de L. On y ajoute 1 suivant si le premier terme $L[0]$ est pair ou pas. Ceci montre l'hérédité.

3. Test de liste croissante :

```
def est_croissante(L: list) -> bool:
  if len(L) <= 1:
    return True
  return L[0] <= L[1] and est_croissante(L[1:])
```

— Terminaison : idem

— Correction : une liste de longueur ≤ 1 est considérée comme croissante, et d'autre part, une liste de longueur ≥ 2 est croissante ssi ($L[0] \leq L[1]$ et $L[1:]$ est croissante).

4. Moyenne :

```
def moyenne(L: list): # liste non vide !
  n = len(L)
  if n==1:
    return L[0]
  return L[0]/n + moyenne(L[1:])*(n-1)/n
```

— Terminaison : idem.
— Correction : Soit $n \geq 2$ et $(a_i)_{1\leq i\leq n}$ une suite finie de réels. Leur moyenne est

$$M\left((a_i)_{1\leq i\leq n}\right) = \frac{1}{n}\sum_{i=1}^{n} a_i = \frac{a_0}{n} + \frac{n-1}{n}\frac{1}{n-1}\sum_{i=2}^{n} a_i = \frac{a_0}{n} + \frac{n-1}{n}M\left((a_i)_{2\leq i\leq n}\right)$$

Ce calcul permet de montrer la correction par récurrence sur la longueur n des listes.

Correction de l'exo ??

1. On a vu en cours une version itérative, voici une version récursive :

```
def fusion(L1: list,L2: list) -> list:
  if L1 == []:
    return L2
  if L2 == []:
    return L1
  if L1[0] <= L2[0]:
    return [L1[0]] + fusion(L1[1:],L2)
  else:
    return [L2[0]] + fusion(L1,L2[1:])
```

2. La complexité est en $O(n)$, avec n la longueur des listes si elles sont de même longueur. (Si elles sont de longueur différente p et q, la complexité est en $O(p+q)$.)
3. Tri fusion :

```
def tri_fusion(L: list) -> list:
  n = len(L)
  if n <= 1:
    return L
  return fusion(tri_fusion(L[:n//2]),tri_fusion(L[n//2:]))
```

Terminaison : si la liste entrée est de longueur ≤ 1 il n'y a pas de nouvel appel récursif, et sinon, il y a deux appels sur des listes de longueur strictement plus petite. L'algorithme termine donc. Correction : preuve par récurrence forte sur la longueur de la liste en entrée.
4. La complexité du tri fusion est en $O(n\log n)$, où n est la taille de la liste en entrée. Pour le démontrer, plusieurs manières :
   — On peut écrire l'arbre de complexité : à chaque appel sur une entrée de taille n, il y a deux appels récursifs sur des entrées de taille $n/2$ (ou $(n\pm 1)/2$ si n est impair), puis il y a de l'ordre de $O(n)$ opérations élémentaires pour fusionner. Le nombre d'opérations élémentaires à chaque niveau de profondeur de l'arbre est toujours de l'ordre de $O(n)$ (avec la même constante !), et la profondeur est $\log n + 1$.
   — Sinon, on peut appliquer le « master theorem » dans sa version améliorée (admise) pour ignorer les parties entières. La complexité du tri fusion est une fonction de n, qui vérifie la relation $T(n) = T(\lfloor n/2\rfloor) + T(\lceil n/2\rceil) + O(n)$. D'après le master theorem, on a $T(n) = O(n\log(n))$.

Correction de l'exercice 5

1. En calculant les premières valeurs on devine que $A(1,n) = n+2$ et que $A(2,n) = 2n+3$. On le montre par récurrence sur n.
2. Pour $n\in\mathbb{N}$, notons $\mathcal{P}(n)$ l'assertion $A(3,n) = 2^{n+3}-3$. Comme $A(3,0) = A(2,1) = 5 = 2^{0+3}-3$, l'assertion $\mathcal{P}(0)$ est vraie. Soit $n\in\mathbb{N}$. Supposons $\mathcal{P}(n)$. Montrons $\mathcal{P}(n+1)$. On a

$$A(3,n+1) = A(2,A(3,n)) = A(2,2^{n+3}-3) = 2(2^{n+3}-3)+3 = 2^{n+1+3}-3.$$

Autrement dit, $\mathcal{P}(n+1)$ est vraie.

3. On a $A(4,0) = A(3,1) = 13$ et $A(4,1) = A(3, A(4,0)) = A(3,13) = 2^{16} - 3$.Ensuite, on a $A(4,2) = A(3, A(4,1)) = 2^{A(4,1)+3} - 3 = 2^{2^{16}} - 3$.
4. On le prouve par récurrence sur m.
   Initialisation : pour $m = 0$, l'assertion est vraie.
   Hérédité : soit $m \in \mathbb{N}$. Supposons $\forall n \in \mathbb{N}, A_m(n) > n$. Montrons $\forall n \in \mathbb{N}, A_{m+1}(n) > n$. On le fait par une deuxième récurrence sur n.

   Initialisation : pour $n = 0$, on a $A_{m+1}(0) = A_m(1) > 1 > 0$.
   Hérédité en n : soit $n \in \mathbb{N}$, supposons $A_{m+1}(n) > n$ et montrons $A_{m+1}(n+1) > n+1$.
   On a alors :

$$\begin{aligned} A_{m+1}(n+1) &= A_m(A_{m+1}(n)) \\ &\geq 1 + A_{m+1}(n) && \text{par hypothèse de récurrence sur } m \\ &\geq 2 + n && \text{par hypothèse de récurrence sur } n \end{aligned}$$

   Ceci conclut la récurrence sur n, dont l'étape d'hérédité sur m.
5. La fonction A_0 est strictement croissante. Soit $m, n \in \mathbb{N}^* \times \mathbb{N}$. Alors

$$A_m(n+1) - A_m(n) = A_{m-1}(A_m(n)) - A_m(n) > 0$$

   d'après la question précédente.
6. Récurrence sur n. Pour $n = 0$, on a bien $A_{m+1}(0) \geq A_m(1)$. Montrons l'hérédité en n. Soit $n \in \mathbb{N}$. Supposons $\forall m \in \mathbb{N}, A(m+1, n) \geq A(m, n+1)$. Montrons $\forall m \in \mathbb{N}, A(m+1, n+1) \geq A(m, n+2)$. Distinguons deux cas. Pour $m = 0$, on a bien $A(1, n+1) \geq A(0, n+2)$ car $(n+1)+2 \geq (n+2)+1$. Soit $m \in \mathbb{N}^*$. Montrons que $A(m+1, n+1) \geq A(m, n+2)$. On a

$$\begin{aligned} A(m+1, n+1) &= A(m, A(m+1, n)) \\ &\geq A(m, A(m, n+1)) && \text{HR et Q5 : croissance de } A_m \\ &\geq A(m, n+1) && \text{Q1 et Q4} \end{aligned}$$

7. Soient $m, n \in \mathbb{N}^2$. Alors :

$$\begin{aligned} A(m+1, n) &\geq A(m, n+1) && \text{Q6} \\ &> A(m, n) && \text{Q1} \end{aligned}$$

8. Version élémentaire de la rédaction :
   0) Pour commencer, $\forall n \in \mathbb{N}$, $A(0, n)$ termine.
   1) Soit $m \in \mathbb{N}$. Supposons que $\forall n \in \mathbb{N}$, $A(m, n)$ termine. Alors $A(m+1, 0)$ termine.
   2) Soit $m \in \mathbb{N}$ et $n \in \mathbb{N}$. Supposons que $\forall n \in \mathbb{N}$, $A(m, n)$ termine et que $A(m+1, n)$ termine. Alors $A(m+1, n+1) = A(m, A(m+1, n))$ termine.
   On effectue alors une double récurrence sur m puis n pour conclure que la fonction termine pour tous $m, n \in \mathbb{N}$.
   (La rédaction plus sophistiquée demande de munir $\mathbb{N} \times \mathbb{N}$ de l'ordre lexicographique $(m, n) \leqslant (m', n') \iff (m < m')$ ou $(m = m'$ et $n \leq n')$. C'est un ordre « bien fondé » au sens de `https://fr.wikipedia.org/wiki/Relation_bien_fond%C3%A9e`. On peut alors faire une récurrence sur cet ensemble ordonné de la même manière que l'on fait une récurrence sur $\mathbb{N}$.)