DS 2 : lundi 8 décembre

1h sans calculatrice

Le candidat encadrera ou soulignera les résultats.
N.B. : le candidat attachera la plus grande importance à la clarté, à la précision et à la concision de la rédaction. Si un candidat est amené à repérer ce qui peut lui sembler être une erreur d'énoncé, il le signalera sur sa copie et devra poursuivre sa composition en expliquant les raisons des initiatives qu'il a été amené à prendre.

Exercice 1 *(SQL : d'après le début de* CCMP *MP-PC-PSI, 2021)*.
Lors de la préparation d'une randonnée, une accompagnatrice doit prendre en compte les exigences des participants. Elle dispose d'informations rassemblées dans deux tables d'une base de données :

— la table `Rando` décrit les randonnées possibles : la clef primaire entière `rid`, son nom `rnom`, le niveau de difficulté du parcours (entier entre 1 et 5) `diff`, le dénivelé (en mètres) `deniv`, la durée moyenne (en minutes) `duree` :

rid	rnom	diff	deniv	duree
1	La belle des champs	1	20	30
2	Lac de Castellane	4	650	150
3	Le tour du mont	2	200	120
4	Les crêtes de la mort	5	1200	360
5	Yukon Ho !	3	700	210
⋮	⋮	⋮	⋮	⋮

— la table `Participant` décrit les randonneurs : la clef primaire entière `pid`, le nom du randonneur `pnom`, son année de naissance `ne`, le niveau de difficulté maximum de ses randonnées `diff_max` et enfin l'identifiant (`rid`) de la dernière randonnée qu'il a effectué `der_rid` :

pid	pnom	ne	diff_max	der_rid
1	Calvin	2014	2	1
2	Hobbes	2015	2	1
3	Susie	2014	2	3
4	Rosalyn	2001	4	4
⋮	⋮	⋮	⋮	⋮

Donner une requête `SQL` sur cette base pour :

1° Compter le nombre de participants nés entre 1999 et 2003 inclus.

2° Calculer la durée moyenne des randonnées pour chaque niveau de difficulté.

3° Extraire le nom des participants accompagné du nom de la dernière randonnée qu'il a effectué

4° Extraire le nom des participants pour lesquels la randonnée n°42 est trop difficile.

5° Extraire les clés primaires des randonnées qui ont un ou des homonymes (nom identique et clé primaire distincte), sans redondance.

Exercice 2 *(suite de Fibonacci et mémoïsation)*.
Écrire la fonction `fibo(n)` qui calcul le n-ième terme de la suite de Fibonacci (ie la suite (F_n) définie par $F_0 = 0$, $F_1 = 1$ et pour tout $n \in \mathbb{N}$, $F_{n+2} = F_{n+1} + F_n$), on procédera de manière récursif et on utilisera de la mémoïsation.

Exercice 3 *(somme maximale de termes consécutif d'une suite)*.
On considère une liste `suite` de $n \geq 2$ entiers relatif $(a_1, \dots, a_n)$, l'objectif est de déterminer la somme maximale de termes consécutifs, dit autrement, si on pose $s_{i,j} = \sum_{k=i}^{j} a_k$, pour $(i,j) \in [\![1,n]\!]^2$ tel que $i \leq j$, on cherche la plus grande valeur prise par $s_{i,j}$.

1° **Approche naïve** .
Une approche brutale est de calculer tous les $s_{i,j}$ et de ne garder en mémoire la plus grande valeur de $s_{i,j}$ rencontrée, par exemple :

```
def somme_max_1(suite):
    n = len(suite)
    maxi = suite[0] # c'est s_{0,0}
    for i in range(n):
        for j in #(A) à compléter
            somme = 0
            for k in #(B) à compléter
                somme = #(C) à compléter
```

```
            if somme>maxi:
                #(D) à compléter
    return maxi
```

(a) Compléter les quatre trous `A`, `B`, `C` et `D`

(b) Donner la complexité de `somme_max_1` en fonction de `n` (nombre d'éléments dans `suite`).

Rappel : $\displaystyle\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$.

(c) (s'il reste du temps à la fin) Proposer une fonction `somme_max_2` qui est une modification de `somme_max_1` mais qui aura une complexité de $O(n^2)$ (on ne détaillera pas le calcul de complexité).

2° **Programmation dynamique**.

Pour $i \in [\![1, n]\!]$, on note M_i la plus grande somme de termes consécutifs de la liste qui se termine en a_i, dit autrement on pose $M_i = \max\limits_{d \in [\![1,i]\!]} \left(\sum\limits_{k=d}^{i} a_k \right)$.

(a) Exprimer la somme maximale de termes consécutif d'une suite en fonction de $M_1, \ldots, M_n$.

(b) Justifier que $f(1) = a_1$ et, pour $i \in [\![1, n-1]\!]$, que $M_{i+1} = \begin{cases} M_i + a_{i+1} & \text{si } M_i \geq 0 \\ a_{i+1} & \text{sinon} \end{cases}$.

(c) Écrire la fonction `somme_max_3` qui permet d'obtenir la somme maximale de termes consécutif de la suite qui aura une complexité en $O(n)$.