DS 4 Mathématiques

Durée : 3 heures. Calculatrices et documents non autorisés.
Les exercices et les problèmes peuvent être traités indépendamment. Dans les problèmes, même s'il est toujours possible de sauter une question, il est bon de prêter attention à la progression des questions et à la structure globale du problème, et de vérifier que les calculs sont corrects avant de passer à la suite.
L'exercice indiqué informatique contient uniquement de l'informatique et sa note sera comptée sur le bulletin dans la matière informatique.

Exercice 1

Déterminer le domaine de définition des fonctions réelles suivantes.

$$f : x \longmapsto \arccos(1 + \ln(x)) \qquad g : x \longmapsto \sqrt{2e^{2x} - 5e^x + 3} \tag{1}$$

$$h : x \longmapsto \frac{1}{\lfloor 2x \rfloor} \qquad i : x \longmapsto \tan(3x) + \sqrt[3]{\arctan(x)} \tag{2}$$

Exercice 2

Dans les exercices de combinatoire, comme d'habitude, on ne cherche pas à donner des valeurs numériques mais on exprime son résultat en fonction de produits, quotients, puissances, factoriels. Par contre, les réponses doivent être justifiées.
Un groupe d'élèves de BCPST s'installe pour son TP en salle informatique contenant 16 postes. On s'intéresse à la question : de combien de façons peut-on installer chacun des élèves à un poste informatique ? Dans chacune des situations suivantes :

1. Il y a 12 élèves. Certains postes seront inoccupés.
2. Il y a 16 élèves. Chacun a droit à sa place seul.
3. Il y a 17 élèves. Deux élèves seront donc nécessairement en binôme sur un même poste.
4. Il y a bien 17 élèves mais un élève nommé X ne veut pas être en binôme.
5. Il y a 17 élèves, et les élèves nommés X et Y peuvent éventuellement être en binôme mais certainement pas ensemble.

Exercice 3

Montrer que l'application suivante est bijective et donner sa bijection réciproque.

$$\begin{aligned} \varphi : \mathbb{R}^3 &\longrightarrow \mathbb{R}^3 \\ (x, y, z) &\longmapsto (x + y + 3z, x - y + 2z, x + 2y - z) \end{aligned} \tag{3}$$

Exercice 4 (informatique)

On cherche à écrire des fonctions Python pour étudier les anagrammes. Pour simplifier, on s'intéresse uniquement aux anagrammes de nombres écrits avec les chiffres de 0 à 9, qu'on représente par la liste de leurs chiffres. Ainsi le nombre 154 est représenté par la liste `[1, 5, 4]`, et un exemple d'anagramme est le nombre 451 représenté par `[4, 5, 1]`. On considère qu'un nombre peut très bien avoir pour premier chiffre 0, ainsi le nombre 2023 représenté par `[2, 0, 2, 3]` a bien 0223 pour anagramme, représenté par `[0, 2, 2, 3]` ; cela ne fait que simplifier le problème.

1. Écrire une fonction `compte(L, x)` qui compte combien de fois le chiffre `x` apparait dans la liste `L` représentant un nombre. En particulier la fonction retourne simplement 0 si le chiffre n'apparait pas.
2. Écrire une fonction `compte_tout(L)` qui retourne une **liste** `C` de longueur 10, où `C[i]` est égal au nombre de fois où apparait le chiffre `i` dans la liste `L`.
3. Écrire une fonction **récursive** `factoriel(n)` qui prend en argument un entier n positif et retourne le nombre $n!$.
4. Compléter la fonction suivante pour qu'elle retourne le nombre d'anagrammes qu'on peut former à partir du nombre représenté par la liste `L`.

```
def nombre_anagrammes(L):
    C = compte_tout(L)
    n = len(L)
    numerateur = ...
    denominateur = ...
    for i in range(10):
        ...
    return numerateur // denominateur # division en nombres entiers
```

5. On souhaite écrire une fonction qui teste si un anagramme donné vérifie la condition qu'il n'y a jamais deux chiffres consécutifs égaux. C'est une fonction qui doit renvoyer la valeur booléenne `True` s'il **n'y a pas** de chiffres consécutifs égaux, et qui doit renvoyer `False` sinon.

   Un élève de BCPST1A écrit la fonction suivante :

```
def non_consecutifs_1A(L):
    for i in range(len(L)):
        if L[i] != L[i+1]: # les deux chiffres consécutifs sont différents
            return True # alors c'est vrai
        else:
            return False # sinon ils sont égaux : c'est donc faux
```

   (a) Quel(s) problème(s) pose cette fonction ? Que renvoie `non_consecutifs_1A([5, 3, 3, 1])` ?

   (b) La corriger et écrire une fonction `non_consecutifs_1B(L)` qui renvoie bien `True` si et seulement si il n'y a pas deux chiffres consécutifs égaux.

6. Que teste la fonction suivante ?

```
def tres_serieux(L, M):
    A = compte_tout(L)
    B = compte_tout(M)
    for i in range(10):
        if A[i] != B[i]:
            return False
    return True
```

Problème

Pour tout entier $n \geqslant 1$ et tout entier $p \geqslant 1$, on cherche à dénombrer les applications surjectives de $[\![1, n]\!]$ dans $[\![1, p]\!]$. Il s'agit aussi du nombre d'applications surjectives de n'importe quel ensemble à n éléments dans un ensemble à p éléments. On le note $S_{n,p}$.

Partie 1

1. En listant simplement les applications de $[\![1, n]\!]$ dans $[\![1, p]\!]$, déterminer les valeurs de $S_{1,1}$, $S_{1,2}$, $S_{2,1}$ et $S_{2,2}$.
2. Que vaut $S_{n,p}$ si $p > n$?
3. Que vaut $S_{n,p}$ si $p = n$?

4. Déterminer $S_{n,1}$.
5. (a) Soit $n \geqslant 1$. Quelles sont les applications de $[\![1,n]\!]$ dans $[\![1,2]\!]$ qui ne sont pas surjectives ?
   (b) En déduire la valeur de $S_{n,2}$ pour tout entier $n \geqslant 1$.

Partie 2

Le but est de démontrer la relation de récurrence (4), que l'on pourra éventuellement admettre pour passer à la partie suivante.
On fixe un couple d'entiers (n,p) tels que $n \geqslant p \geqslant 2$ et on désigne par $\mathscr{S}$ l'ensemble des surjections de $[\![1,n]\!]$ dans $[\![1,p]\!]$. On exprimera les réponses aux questions 6b et 7b en fonction de p, $S_{n-1,p}$ et $S_{n-1,p-1}$.

6. (a) Soit $\varphi \in \mathscr{S}$. On pose $\beta = \varphi(1)$ et on suppose que β admet un seul antécédent par φ. Justifier que l'application $\psi_1 : [\![2,n]\!] \to [\![1,p]\!] \setminus \{\beta\}$, $x \mapsto \varphi(x)$ est surjective.
   (b) En déduire une expression du cardinal de $\mathscr{S}_1 = \{\varphi \in \mathscr{S} \mid \text{Card}\{x \in [\![1,n]\!] \mid \varphi(x) = \varphi(1)\} = 1\}$.
7. (a) Soit $\varphi \in \mathscr{S}$. On pose $\beta = \varphi(1)$ et on suppose que β admet au moins deux antécédents par φ. Justifier que l'application $\psi_2 : [\![2,n]\!] \to [\![1,p]\!]$, $x \mapsto \varphi(x)$ est surjective.
   (b) En déduire une expression du cardinal de $\mathscr{S}_2 = \{\varphi \in \mathscr{S} \mid \text{Card}\{x \in [\![1,n]\!] \mid \varphi(x) = \varphi(1)\} \geqslant 2\}$.
8. Montrer que :
$$S_{n,p} = p(S_{n-1,p} + S_{n-1,p-1}). \tag{4}$$

Partie 3

On applique maintenant la relation (4).

9. Écrire une fonction Python **récursive** `surjections(n, p)` qui calcule $S_{n,p}$ par cette formule de récurrence, pour tous $n \geqslant 1$ et $p \geqslant 1$.
10. Pensez-vous que la fonction ainsi écrite va être rapide à calculer pour l'ordinateur ? Justifier.
11. Démontrer la formule, pour $n \geqslant 1$ et $p \geqslant 1$
$$S_{n,p} = (-1)^p \sum_{k=1}^{p} (-1)^k \binom{p}{k} k^n. \tag{5}$$
soigneusement par récurrence en posant l'hypothèse
$$P(n) : \qquad \forall p \geqslant 1, \quad S_{n,p} = (-1)^p \sum_{k=1}^{p} (-1)^k \binom{p}{k} k^n \tag{6}$$
12. En déduire une fonction Python itérative (avec des boucles, mais pas de récursivité) `surjections2(n, p)` qui calcule $S_{n,p}$. On supposera qu'on dispose déjà d'une fonction `binome(k, n)` qui calcule le coefficient binomial $\binom{n}{k}$.