Interrogation d'Informatique

du 19 au 23/1/2026

Nom :		Prénom :	

Consignes

- En Mathématiques, les énoncés du cours doivent être complètement écrits : un cadre, des hypothèses et une conclusion.
- En Informatique : les scripts doivent être correctement indentés, en mettant en valeur l'indentation à l'aide d'une barre verticale.
- La note finale tiendra compte, directement ou indirectement, de la qualité de la rédaction et de la présentation.
- **Le crayon à papier ne sera pas corrigé.**
- **L'usage de la calculatrice est interdit.**

/12

RÉCURSIVITÉ

Exercice 1 | Proche du cours /3 Écrire une fonction **récursive** d'en-tête `factor(n)`, qui prend en paramètre un entier n et qui renvoie $n!$.

Exercice 2 | Fonction mystère [Solution] On considère le programme ci-après.

```
def mystere(n):
    if n == 0:
        return 0
    else:
        return n**2 + mystere(n-2)
```

1. /1 Que renvoie `mystere(0)` ? Expliquez.

2. /1 Que renvoie `mystere(4)` ? Expliquez.

3. /1 Que dire de l'appel `mystere(3)` ? Expliquez.

DICTIONNAIRES

Exercice 3 | [Solution] On considère la fonction ci-après (`txt` étant une chaîne de caractères quelconque).

```
def inconnue(txt):
    D = {}
    n = len(txt)
    for i in range(n):
        x = txt[i]
        if x not in D:
            D[x] = [i]
```

```
        else:
            D[x].append(i)
    return D
```

1. /2 Donner les différents contenus de la variable `D` lors de l'exécution de `inconnue("bonbon")`.

2. /1 Conjecturer, avec un vocabulaire précis, ce que renvoie cette fonction de façon générale.

3. /2 En utilisant la fonction précédente, en déduire une fonction d'en-tête `nb_occur(x, txt)` qui étant donnée une chaîne `txt` renvoie le nombre de fois que la chaîne `x` (de longueur 1) apparait dans `ch`. Par exemple `nb_occur("o", "quelbonbonbon")` renverra 3.

4. /1 En utilisant la fonction précédente, expliquer comment obtenir le nombre de fois qu'une lettre `x` apparait dans un texte `txt`.

Solution (exercice 2) [Énoncé]

1. `mystere(0)` renvoie 0, c'est un cas terminal.
2. `mystere(4)` → `16+mystere(2))` → `16+4+inconnue(0))` → $\boxed{20}$.

```
>>> mystere(4)
20
```

3. `mystere(3)` ne s'arrête pas; car appelle `mystere(1)` puis `mystere(-1)` *etc.*. On arrive donc jamais sur un cas terminal, et ainsi une erreur de dépassement de taille de pile sera affichée.

Solution (exercice 3) [Énoncé]

1. Résumons dans un tableau les différentes exécutions de la variable `D` lors de l'appel `inconnue("bonbon")`. L'entier `k` désigne le numéro de l'itération, avec pour convention que $k = 0$ correspond à l'initialisation.

k	i_k	D_k
0		{}
1	1	{"b" : [0]}
2	2	{"b" : [0], "o" : [1]}
3	3	{"b" : [0], "o" : [1], "n" : [2]}
4	4	{"b" : [0, 3], "o" : [1], "n" : [2]}
5	5	{"b" : [0, 3], "o" : [1, 4], "n" : [2]}
6	6	{"b" : [0, 3], "o" : [1, 4], "n" : [2, 5]}

2. La fonction renvoie un dictionnaire de clefs les lettres (sans répétition) d'une chaîne, et de valeurs la liste des indices où la lettre apparait dans la chaîne.
3.
```
def nb_occur(x, txt):
    D = inconnue(txt)
    return len(D[x])
```

```
>>> nb_occur("o", "quelbonbonbon")
3
```

4. On calcule simplement `nb_occur(x, txt)`.

```
>>> txt = "bonbon"
>>> nb_occur("b", txt)
```