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.**

/16

RÉCURSIVITÉ

Exercice 1 | [Solution] On considère la fonction suivante :

```
def inconnue(a, b, c):
    if b == 0:
        return c
    else:
        return inconnue(a, b-1, c+a)
```

1. /1 Que renvoie l'instruction `inconnue(4, 3, 0)` ?

2. 2.1) /1 Conjecturer, et expliquer, ce que renvoie l'instruction `inconnue(n, m, 0)`.

2.2) /2 Écrire une fonction itérative `f(n, m)` et renvoyant le même résultat que `inconnue(n, m, 0)`. La fonction ne devra pas utiliser le symbole `*`, c'est-à-dire le symbole de multiplication Python.

TRIS

Exercice 2 | Proche du cours

1. /3 Écrire une fonction d'en-tête `tri_test_croissant(L)`, qui prend en paramètre une liste `L` et qui renvoie `True` si elle est triée par ordre croissant, et `False` sinon.

2. /3 Compléter la fonction ci-après pour qu'elle renvoie une nouvelle liste triée en ordre croissant selon le tri rapide.

```
def tri_rapide_rec(L):
    if ________________________:
        return L
    else:
        pivot = L[0]
        inf_pivot = []
        sup_pivot = []
        for x in L[1:]:
            if ________________________:
                inf_pivot.append(x)
            else:
                ________________________
        return tri_rapide_rec(________________________) + \
        ↪ _____________ + \
        ↪ tri_rapide_rec(________________________)
```

/1 Expliquer précisément quelle partie modifier pour avoir une liste triée dans l'ordre décroissant.

DICTIONNAIRES

Exercice 3 | Trafic de notes [Solution] On suppose construit un dictionnaire du type ci-dessous :

```
Notes = {"Jean DUPONT": ["1BC1", 8.3], "Marie BOSSARD": ["1BC2", 13.0], ...}
```

contenant les notes des deux classes de 1BC d'un célèbre lycée bordelais. Les clefs sont donc des chaînes contenant prénom, suivi d'un espace, et nom de l'étudiant(e), les valeurs sont des listes contenant deux objets : le premier est une chaîne contenant la classe (`"1BC1"` ou `"1BC2"`), le second est la note obtenue.

1. /1 On suppose que `Harry POTTER` ne fait pas encore partie des élèves. Écrire une instruction permettant d'ajouter `Harry POTTER` en `1BC1`, avec la note de `19.0`.

2. /1 Écrire une instruction permettant de supprimer `Jean DUPONT` (et sa note/classe) du dictionnaire.

3. /3 Un professeur peu scrupuleux décide d'ajouter 1 point à tous les 1BC1 et de retirer 1 point à tous les 1BC2.
Écrire une fonction `trucage_notes(D)` qui prend en argument un dictionnaire `D` du même type que `Notes`, et qui modifie en conséquence le dictionnaire. *On notera donc cette fonction modifiera le dictionnaire `D` passé en argument mais ne renverra rien. Elle agit donc par « effets de bord ». On ne souciera pas des notes éventuellement négatives et dépassant 20.*

SOLUTIONS DES EXERCICES

Solution (exercice 1) [Énoncé]

1. `inconnue(4, 3, 0)` → `inconnue(4, 2, 4))` → `inconnue(4, 1, 8))` → `inconnue(4, 0, 12))` = $\boxed{12}$.
2. **2.1)** `inconnue(n, m, 0)` → `inconnue(n, m-1, n)` → `inconnue(n, m-2, 2n)` → `inconnue(n, m-3, 3n)` → ⋯ → `inconnue(n, 0, mn)` = $\boxed{m \times n}$. La variable b sert alors ici de « compteur récursif ».

   2.2)
```
def f(n:int, m:int)->int:
    S = 0
    for _ in range(m):
        S += n
    return S
```
```
>>> f(4, 3)
12
```

Solution (exercice 3) [Énoncé]

```
>>> Notes = {"Jean DUPONT": ["1BC1", 8.3], "Marie BOSSARD": ["1BC2", 13.0]}
```

1.
```
>>> Notes["Harry POTTER"] = ["1BC1", 19.0]
>>> Notes
{'Jean DUPONT': ['1BC1', 8.3], 'Marie BOSSARD': ['1BC2', 13.0], 'Harry POTTER': ['1BC1', 19.0]}
```

2.
```
>>> del Notes["Jean DUPONT"]
>>> Notes
{'Marie BOSSARD': ['1BC2', 13.0], 'Harry POTTER': ['1BC1', 19.0]}
```

3.
```
def trucage_notes(D:dict)->None:
    for etudiant in D:
        classe = D[etudiant][0]
        if classe == "1BC1":
            D[etudiant][1] += 1
        else:
            D[etudiant][1] -= 1
```
```
>>> Notes
{'Marie BOSSARD': ['1BC2', 13.0], 'Harry POTTER': ['1BC1', 19.0]}
>>> trucage_notes(Notes)
>>> Notes
{'Marie BOSSARD': ['1BC2', 12.0], 'Harry POTTER': ['1BC1', 20.0]}
```