

# **BACCALAUREAT**

**SESSION 2022**

---

**Épreuve de l'enseignement de spécialité**

## **NUMERIQUE et SCIENCES INFORMATIQUES**

**Partie pratique**

**Classe Terminale de la voie générale**

---

**Sujet n°09**

---

**DUREE DE L'ÉPREUVE : 1 heure**

**Le sujet comporte 3 pages numérotées de 1 / 3 à 3 / 3  
Dès que le sujet vous est remis, assurez-vous qu'il est complet.**

*Le candidat doit traiter les 2 exercices.*

### EXERCICE 1 (4 points)

Soit un nombre entier supérieur ou égal à 1 :

- s'il est pair, on le divise par 2 ;
- s'il est impair, on le multiplie par 3 et on ajoute 1.

Puis on recommence ces étapes avec le nombre entier obtenu, jusqu'à ce que l'on obtienne la valeur 1.

On définit ainsi la suite  $(u_n)$  par

- $u_0 = k$ , où  $k$  est un entier choisi initialement ;
- $u_{n+1} = u_n / 2$  si  $u_n$  est pair ;
- $u_{n+1} = 3 \times u_n + 1$  si  $u_n$  est impair.

**On admet que, quel que soit l'entier  $k$  choisi au départ, la suite finit toujours sur la valeur 1.**

Écrire une fonction `calcul` prenant en paramètres un entier  $n$  strictement positif et qui renvoie la liste des valeurs  $u_n$ , en partant de  $k$  et jusqu'à atteindre 1.

Exemple :

```
>>> calcul(7)
```

```
[7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1]
```

### EXERCICE 2 (4 points)

On affecte à chaque lettre de l'alphabet un code selon les tableaux ci-dessous :

A	B	C	D	E	F	G	H	I	J	K	L	M
1	2	3	4	5	6	7	8	9	10	11	12	13

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
14	15	16	17	18	19	20	21	22	23	24	25	26

Pour un mot donné, on détermine d'une part son *code alphabétique concaténé*, obtenu par la juxtaposition des codes de chacun de ses caractères, et d'autre part, son *code additionné*, qui est la somme des codes de chacun de ses caractères. Par ailleurs, on dit que ce mot est « *parfait* » si le code additionné divise le code concaténé.

Exemples :

- Pour le mot "PAUL", le code concaténé est la chaîne 1612112, soit l'entier 1 612 112.

Son code additionné est l'entier 50 car  $16 + 1 + 21 + 12 = 50$ .

50 ne divise pas l'entier 1 612 112 ; par conséquent, le mot "PAUL" n'est pas parfait.

- Pour le mot "ALAIN", le code concaténé est la chaîne 1121914, soit l'entier 1 121 914.

Le code additionné est l'entier 37 car  $1 + 12 + 1 + 9 + 14 = 37$ .

37 divise l'entier 1 121 914 ; par conséquent, le mot "ALAIN" est parfait.

Compléter la fonction `est_parfait` ci-dessous qui prend comme argument une chaîne de caractères `mot` (en lettres majuscules) et qui renvoie le code alphabétique concaténé, le code additionné de `mot`, ainsi qu'un booléen qui indique si `mot` est parfait ou pas.

```
dico = {"A":1, "B":2, "C":3, "D":4, "E":5, "F":6, "G":7, \
        "H":8, "I":9, "J":10, "K":11, "L":12, "M":13, \
        "N":14, "O":15, "P":16, "Q":17, "R":18, "S":19, \
        "T":20, "U":21, "V":22, "W":23, "X":24, "Y":25, "Z":26}

def est_parfait(mot) :
    #mot est une chaîne de caractères (en lettres majuscules)
    code_c = ""
    code_a = ???
    for c in mot :
        code_c = code_c + ???
        code_a = ???
    code_c = int(code_c)
    if ??? :
        mot_est_parfait = True
    else :
        mot_est_parfait = False
    return [code_a, code_c, mot_est_parfait]
```

Exemples :

```
>>> est_parfait("PAUL")
[50, 1612112, False]
>>> est_parfait("ALAIN")
[37, 1121914, True]
```

```
from doctest import testmod

def calcul(n):
    """Définition d'une suite.
    Args:
        n (int): nombre entier de départ

    Returns:
        list: tableau contenant toutes les étapes de calcul

    Tests et Exemples:
    >>> calcul(7)
    [7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1]
    """
    liste = []
    u0 = n
    while u0 != 1:
        # condition d'arrêt : u0 == 1
        liste.append(u0)

        # détermination du prochain terme
        if u0 % 2 == 0:
            u0 = u0 // 2
        else:
            u0 = 3 * u0 + 1

    # ajout de la dernière valeur '1' qui a
    # terminée la boucle
    liste.append(u0)
    return liste

assert calcul(7) == [7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1]
testmod()
```

```
dico = {"A":1, "B":2, "C":3, "D":4, "E":5, "F":6, "G":7, \
        "H":8, "I":9, "J":10, "K":11, "L":12, "M":13, \
        "N":14, "O":15, "P":16, "Q":17, "R":18, "S":19, \
        "T":20, "U":21, "V":22, "W":23, "X":24, "Y":25, "Z":26}

def est_parfait(mot) :
    #mot est une chaîne de caractères (en lettres majuscules)
    code_c = ""
    code_a = 0
    for c in mot :
        code_c = code_c + str(dico[c])
        code_a = code_a + dico[c]
    code_c = int(code_c)
    if code_c % code_a == 0 :
        mot_est_parfait = True
    else :
        mot_est_parfait = False
    return [code_a, code_c, mot_est_parfait]

assert est_parfait("PAUL") == [50, 1612112, False]
assert est_parfait("ALAIN") == [37, 1121914, True]

from random import randint, choice
prenom = ''.join([choice(['B', 'D', 'F', 'H']), choice(['B', 'D', 'F', 'H']), choice
(['B', 'D', 'F', 'H'])])
print(est_parfait(prenom))
```